Software Entrepreneurship: course notes

Rose, Jeremy

Publication date:
2012

Document Version
Early version, also known as pre-print

Software Entrepreneurship: course notes, Department of Computer Science, Aalborg University

Jeremy Rose

Prepared with sponsorship from>

Acknowledgements: Aalborg University Computer Science seventh semester students for help in preparing these notes.

Aalborg University

Department of Computer Science

Selma Lagerløfs Vej 300

Aalborg 9220

Denmark

Contents

Entrepreneurship and software professionals

Every great company started as a new venture, and every great product started as a germ of an idea in someone's imagination. By and large we all have good ideas, but relatively few of us have the drive, energy and skill to promote and develop those ideas and to see them through to commercial fruition. Entrepreneurship is the science and the art of setting new ventures in motion. Why should you, as a software (or IT) professional know something about entrepreneurship? The answer has three parts:

1. Society wants you to be entrepreneurial. Entrepreneurial ventures, particularly high-growth ventures such as knowledge and technology ventures, are thought to stimulate commerce and produce new jobs - a classic example in Denmark is the wind technology industry. As software professionals, you are in a position to be excellent wealth generators - not only for your own private economy, but also for the companies you work for and the society you live in.
2. Your employer wants you to be entrepreneurial. They don't necessarily want you to quit and form your own spin off company in direct competition with them, but software companies are dependent on innovation and growth for survival and it is the entrepreneurial spirits (intrapreneurs) that drive this.
3. Do you want to work on someone else's project all your life? As working professionals you need interesting, stimulating and productive working lives; this means at various points in your life you will want to set different kinds of ventures in action. Some of you (not many) will create your own businesses, but all of you will have ideas for software, projects, and practices that you will want to promote in your work environments.

Entrepreneurship is a relatively young discipline with its roots in management theory and economics; however there is a considerable literature and the subject figures on most university curricula. Nevertheless the things that software professionals need to know about entrepreneurship and the competences that they need to acquire are not necessarily generalized skills and knowledge that everyone from carpenters to hairdressers to communication consultants need. These notes therefore identifiy some specific skills and knowledge that are relevant for software and IT professionals. Whereas entrepreneurship is often taught in a practical way (how do you develop your idea for a business into a business plan), these are mainly theoretical course notes. They set out to answer the question: what should you know and understand, rather than how do you develop a new venture.

We'll understand entrepreneurship as driving (promoting) a software or IT venture, where the venture might be a new business, software product, service or engineering practice, and the context might be within an existing firm or the creation of a new one. However this relatively wide focus area won't disguise the fact that there's a classical or traditional central area of concern to the subject - the software or IT start-up company. What should you look for in this kind of start-up? Cusumano answers:

- a strong management team
- an attractive market
- a compelling new product, service or hybrid solution (mixture of both)
- strong evidence of customer interest

- a plan to overcome the 'credibility gap' (the fear among customers and investors that the start-up will fail)
- a business model showing early growth and profit potential
- flexibility in strategy and product offerings (the ability to adapt to changing circumstances)
- the potential for a large payoff to investors.

As with all traditional wisdoms, there are some good reasons why we should look somewhat critically at this advice. There is a rather glamorous world of venture capital and Silicon Valley, which you can read about in Cusumano's 'The Business of Software' [1] and Kaplan's 'Start Up' [2] but here your survival chances are not good. Kaplan's award-winning Go Corporation and their pen operating system ended, after $75million and six years in a car boot sale, as do about 60% of high tech companies with venture capital funding [1]. A further 30% end up in mergers and liquidations. We'll try to keep our feet more firmly on the ground, and we'll also look at realistic ways of promoting new ventures with very limited resources; sometimes consisting only of your software competences, a good product idea and a lot of drive and energy.

Research method

How are the notes put together? It's a literature survey. This means that it is a thematic account of the relevant published scientific research. Books and articles are identified by querying the known bibliographical databases with the search terms 'software,' 'information technology' and 'entrepreneurship,' together with some related terms to ensure completeness. A complete list of sources is given at the end of the notes. In order to be considered, research contributions must consider both themes: entrepreneurship and software (or IT); in other words we are interested in literature which is focused on the creation of software ventures, rather than general literatures about entrepreneurship, IT management or software development. Some other explanatory and background ideas are taken from the wider entrepreneurship and business literature. The material in the notes is tested with the help of my computer science students; they have helped to pick and summarise material which is relevant to them, and reviewed the early drafts. There's a list of relevant sources (or further reading) at the end of each chapter, and a complete list of references at the end of the notes.

Two paradigms, four major themes

Within our broad understanding of the topic of software entrepreneurship, the notes are organised under two paradigms and four major themes. The two paradigms are explained a little later; the major content of the notes are organised under these headings:

- The Software Start-up takes the most classical view of the subject: here we study starting a software firm.
- E(M)-entrepreneurship (entrepreneurship for electronic and mobile commerce) investigates developing a business idea which is software-dependent or software–intensive. These are often businesses that exist mainly on the net, or operate through various forms of mobile services. Examples are Facebook and internet gambling. In a software dependent business the central value proposition is not the construction of software, however the business model is dependent on tailored software for its execution - this makes it interesting for software professionals.

- Open entrepreneurship looks at a special form of entrepreneurship made possible by the open source movement, the existence of free software, and value generation based on open business models. It's a way of getting started that is becoming more common; current developments in the software industry suggest that it is likely to be more prevalent in the future.
- Intrapreneurship involves promoting a new software product, service or practice from within an existing software firm.

Software and entrepreneurship: some formative questions

What should a software or IT professional know and understand about entrepreneurship? In order to begin our exploration of these topics we'll formulate some questions. These represent some relevant areas that we don't necessarily expect to know through our professional experience, or business or engineer educations.

Software (IT applications) and value

Whereas engineers are good at understanding how to build software, and IT managers good at organizing IT services in companies and government institutions, here we will need to understand the value of software. This is because our fundamental job as entrepreneurs is to create value. Our venture must be valuable to our customers and stakeholders otherwise they will not support it by investing in it or by buying our products and services. Therefore we need to understand

- what forms of value can software create?

The value in software is normally associated with knowledge and knowledge is a form of intellectual property which can be protected by copyright and other mechanisms. Software not protected by copyright can be freely copied, which diminishes its value for its developers, so we should ask

- how do you recognise your intellectual property in your software work and protect it?

You can also (paradoxically) create value whilst giving your source code away, so

- how do you leverage open source software and the open source development model?

Software business models

The value of IT applications is articulated in business models – these specify how services and products are translated into value which can earn income (revenues). We need to understand

- how do you make a business out of a software product idea?

There are many kinds of businesses which create value from software; some of them are software development companies, but others have different business models which are highly dependent on software. Think of Facebook (an online social network supported by advertising) and eBay (an online auction house supported by micro transaction fees). We also need to know

- what business models do young software and software-dependent firms use?

If we build software we usually need to understand our users, but if we create value we need to understand our customers, collectively known as our market(s), so

- how do you commercialize and market a new product?
- how do you understand the market you are in, or create a market if it does not exist?

Creating IT applications is expensive – many developer hours - so we will need to answer the question

- how do you find the necessary resources to develop your firm or product idea?

IT and software start-up management

Young IT professionals are usually good and building software, and some have managed a few projects, but the demands of running a small company are many and varied. Salaries must be paid, the law followed, customers found, services advertised and someone must sweep the office floor. We should ask

- who becomes a software entrepreneur, and what skills and capabilities do you need?

and

- how do you manage a start-up?

In particular, a young company will need a development practice that works for them, and this will probably not be the complex methodology and software engineering principles familiar from large companies, so

- what engineering and development practices do you need to develop software in a very young firm?

Experience shows that the environment that young companies operate in is rather complex, with many stakeholders: we should understand:

- how do you manage competitors and customers?

and

- how do young software firms grow through networking and partnering?

Many software companies get started through developing new technologies or finding new twists to existing ones, and we need to know

- what is the role of innovation in software entrepreneurship?

After the start-up

The survival rate for young IT companies is not good, and a very small company usually wants to become a medium size company to be able to stabilise, attract investment and develop new products and services; her we should know

- how does a software start-up survive and grow?

and

- why do software ventures fail?

You like your existing job

You work in an interesting position in a good company and you don't need to start a new company, you just need to get your colleagues (or maybe the directors) to jump on your idea. You have an idea for a new software product or IT application, or you would like to change your development practice (maybe by introducing an agile method such as SCRUM or XP). Now you need to understand

- how do you promote a software product or practice from within a software development firm?

These and other questions form the basis for the coming investigation and we will return to them in the conclusion.

Sources

CUSUMANO, M. (2004) The business of software: What every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad, New York, Free Press.

KAPLAN, J. (1994) Start Up, London, Warner.

The theoretical landscape: some reference theories

In this chapter we will briefly introduce some reference theories. Reference theories are not the central ideas of software entrepreneurship, but some of the theories that are commonly referred to by the science. It follows that you can't understand software entrepreneurship science without some knowledge of these. We'll focus on some here which run though many of the issues discussed and introduce some others as they are appropriate.

The first group of theories concentrate on how businesses generate value, and how they use their internal resources to generate sustainable competitive advantage. Software entrepreneurs need to understand something of how theorists frame the idea of a successful business, because that is what they are trying to create.

Business model

Business models contribute to software entrepreneurship by helping us think about software ventures in a business-oriented way. A business model, according to Chesbrough [3]:

- articulates the value proposition of a business (how it generates value for its customers)
- identifies a market segment (the group or groups of customer users who will pay for the product and use it)
- defines the structure of the firm's value chain (see below)
- specifies the revenue generation mechanisms (describes what costs should be incurred and revenues earned)
- describes the position of the firm within the value network (how it interacts with its suppliers and customers)
- formulates the competitive strategy (how it will remain profitable whist competing with other firms with similar products)

Business models are sometimes descriptive (explaining how a company works), but Osterwalder [4] introduces the idea of a business model canvas as a design tool. The business model canvas encourages an entrepreneur to think about the following areas:

- value propositions (what value do we deliver to the customer? which of our customer's problems are we helping to solve? what bundles of products and services are we offering to each customer segment? which customer needs are we satisfying?)
- customer segments (for whom are we creating value? who are our most important customers?
- customer relationships (what type of relationship does each of our customer segments expect us to establish and maintain with them? which ones have we established? how are they integrated with the rest of our business model? how costly are they?)
- channels (through which channels do our customer segments want to be reached? how are we reaching them now? how are our channels integrated? which ones work best? which ones are most cost-efficient? how are we integrating them with customer routines?)
- key activities (what key activities do our value propositions require? our distribution channels? customer relationships? revenue streams?)

- key resources (what key resources do our value propositions require? our distribution channels? customer relationships? revenue streams?
- key partners (who are our key partners? who are our key suppliers? which key resources are we acquiring from partners? which key activities do partners perform?)
- cost structure (what are the most important costs inherent in our business model? which key resources are most expensive? which key activities are most expensive?)
- revenue streams (for what value are our customers really willing to pay? for what do they currently pay? how are they currently paying? how would they prefer to pay? how much does each revenue stream contribute to overall revenues?) [4]

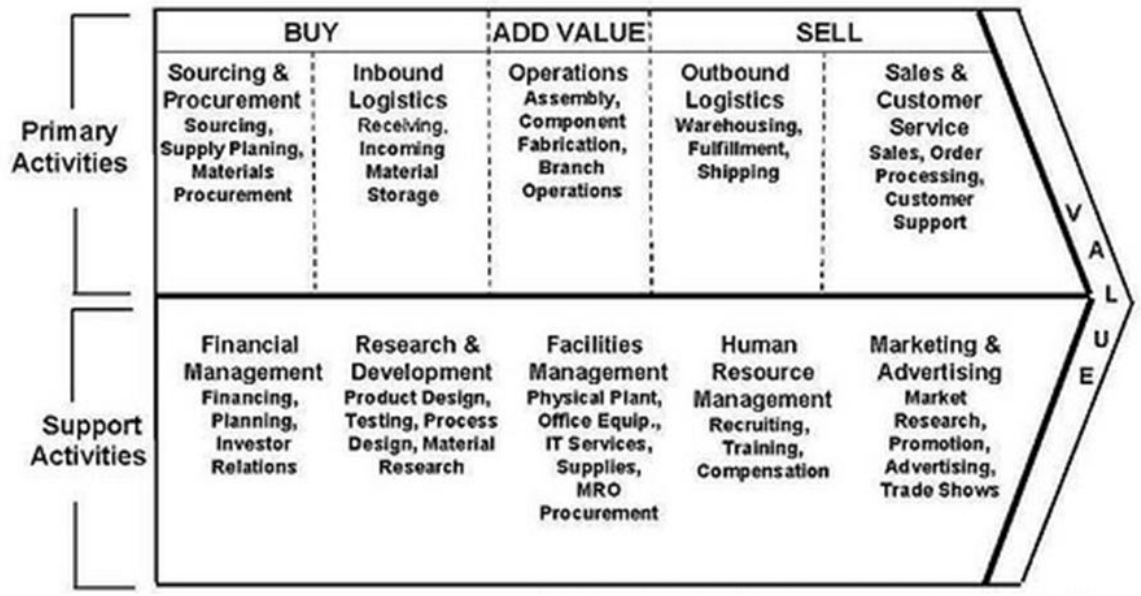
You can sketch the central ideas of how a business will function by working with the canvas (for example on post-it stickers distributed round a whiteboard). As engineers and IT managers we have a wider focus (including the design and programming of software products) so we will think of a business model as the business logic behind a software venture. It's not an engineering design for what we want to do, but the business behind the engineering. It will also make sense to understand the business behind a new software product or software engineering practice that we wish to promote.

Generic business models help explain types of businesses (bricks and clicks, direct sale, franchise etc.). We'll investigate some software business models later.

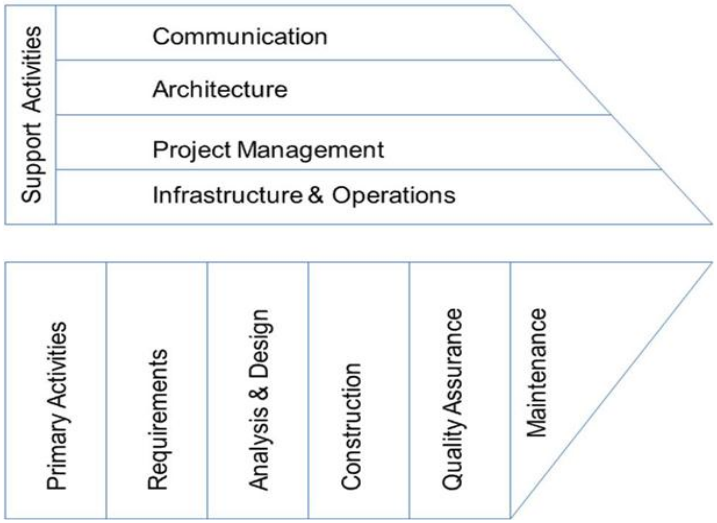
Several theories (next) contribute by focusing our attention on how a firm generates value, and creates a sustainable competitive advantage over rival firms.

Value Chain

A firm can be understood as a collection of value-adding activities (Porter [5]), taking inputs such as raw materials, energy, and human labour and transforming them so that they can later be sold at a profit.



Imagine a car manufacturer: various kinds of resources and ready-made parts come from different sources, are worked and assembled in a production line, and then sold to distributors. These primary activites are made possible by support activies, which don't contribute directly to the creation of value, but enable it. Primary and support activities create value, which can be expressed in monetary terms when customes by cars. Manufacturing industry is easily portrayed this way, but software companies don't usually have material inputs in the same way. Nevertheless, a software firm (for instance a traditional consultancy house), understood as a collection of value-adding activities could be understood something like this:



The primary activities of a traditionally organised software consultancy can be understood as requirements gathering, analysis and design, construction (programming is the fundamental value-adding activity, since it is the code that can later be sold), test and maintenance. A variety of support activities make the primary activities possible. If one takes a purely economic perspective, both primary and support activities cost money; and the value added is the difference between the revenues generated from code and services, and the cost of primary and support activities.

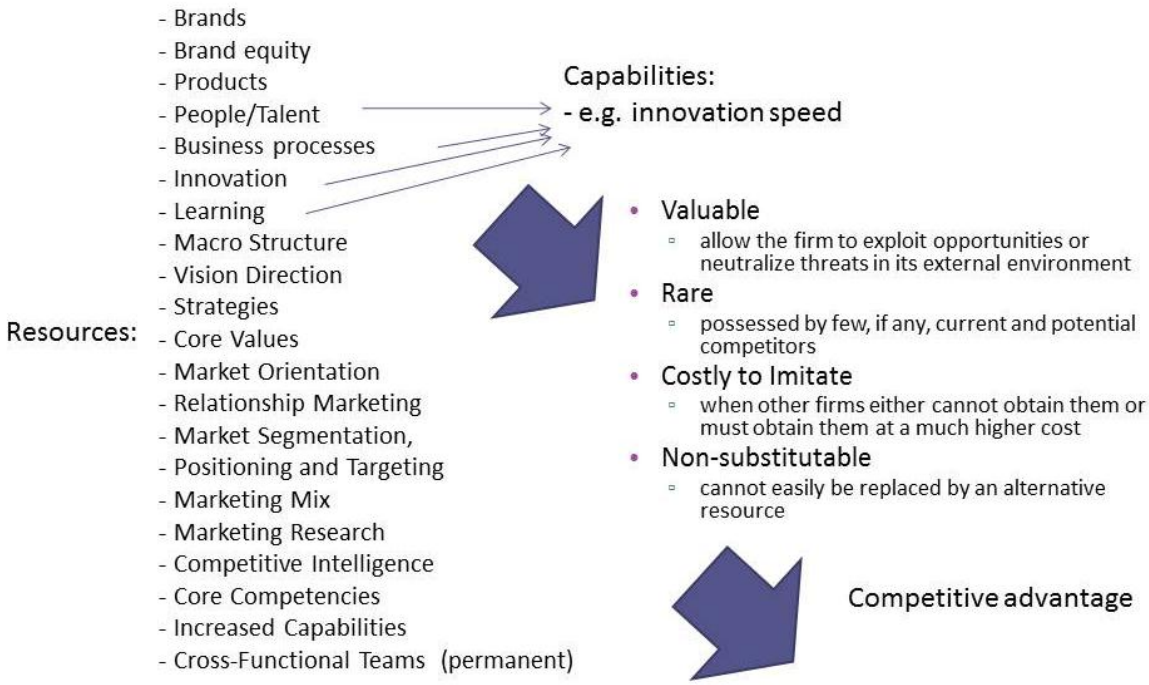
Resource-based theory of the firm

A different understanding of a firm is as a collection of resources and capabilities (things the firm can do especially well). This is known as the resource-based theory of the firm[6]. Resources can cover employees, offices, machines and products, but also less tangible things such brand (Apple, Google), vision and market research. Where resources are appropriately combined, they make organisational capabilities. If an organisation combines some talented people and learning resources with some funding for innovation, then they might have an organisational capability to innovate quickly. Where those resources and capabilities are:

- valuable (allow the firm to exploit opportunities or neutralize threats in its external environment)
- rare (possessed by few, if any, current and potential competitors)

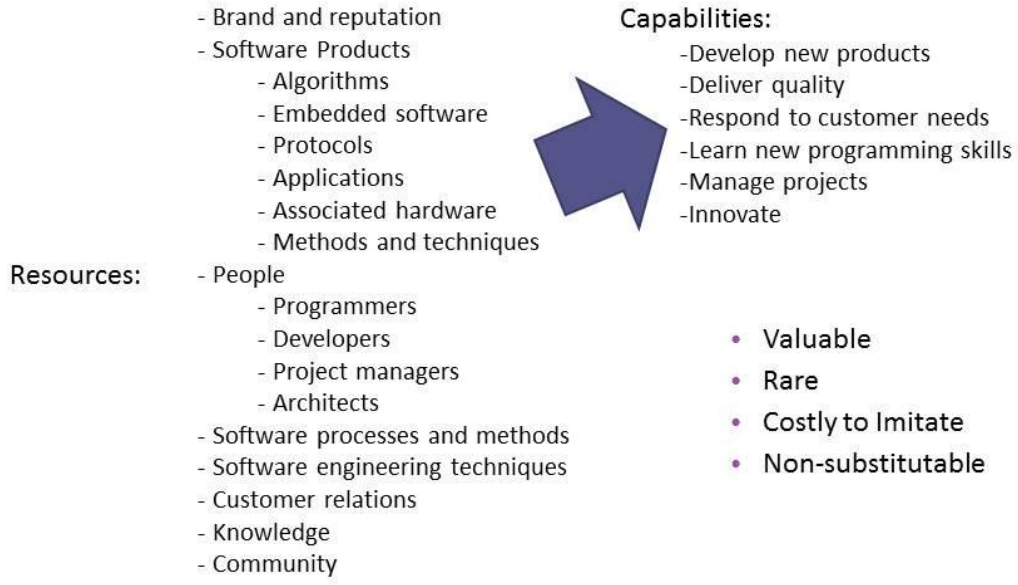
- costly to imitate (when other firms either cannot obtain them or must obtain them at a much higher cost) and/or
- non-substitutable (cannot easily be replaced by an alternative resource)

they can lead to competitive advantage and thus survival in a particular industry and market



Dollinger[7] uses the resource-based theory of the firm as the starting point of his account of entrepreneurship.

If we think about the software and IT industries, then some resources and capabilities could look like this:



If these resources added up to a capability to (for example) write effective encryption algorithms, then these might be valuable for some customers, rare, costly to imitate and difficult to find replacements for.

Knowledge-based theory of the firm

A more attractive, but less well understood way of thinking about a software company is through the knowledge-based theory of the firm[8]. Software is intangible - only the hardware on which it runs and is created is a physical resource – so it can easily be understood as the product or combination of different kinds of knowledge: about programming, engineering techniques, algorithms, methods, project management, users and their domains, markets, and so on. The company that is able to combine these different forms of knowledge in a productive way writes software that is valuable. Knowledge-based companies:

- work proactively with knowledge and learning
- create, transfer and transform knowledge into lasting competitive advantage
- develop an organisational learning culture
- understand the benefits and different uses of both tacit (internalised) and explicit knowledge
- understand that sticky (difficult to transfer, internalised) knowledge is valuable, rare, hard to imitate, and hard to substitute.

Core competences

Hamel and Prahalad[9] argue that what makes a business successful is its ability to concentrate on, and develop its core competences. Core competences

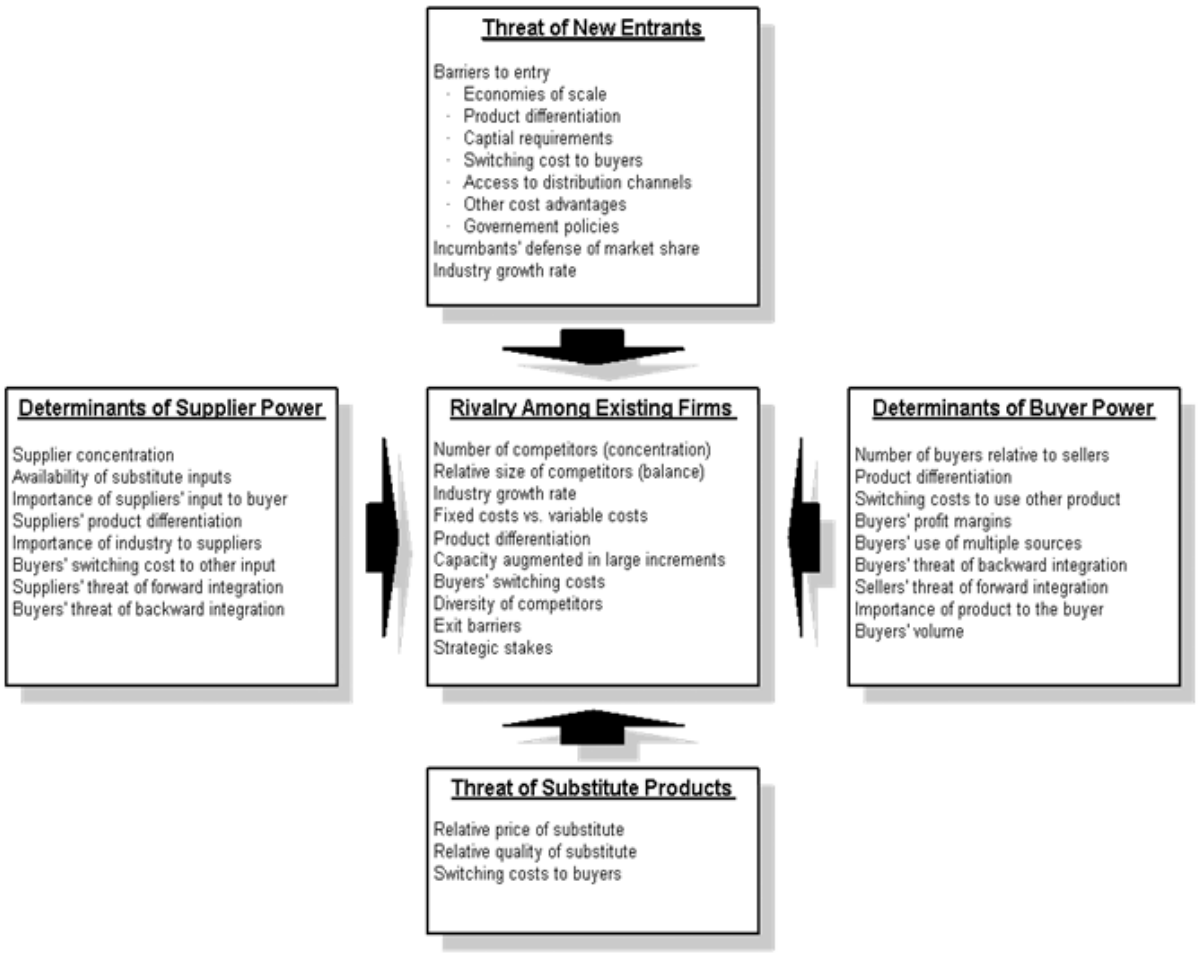
- make a significant contribution to customer perceived value or to the financial health of the organization.
- are unique or performed in a way that is substantially superior to competitors.
- are capable of being applied to new products and services
- provide access to a variety of markets
- contribute strongly to products and services
- are difficult to imitate

Other, less important aspects of the business can be bought in or out-sourced. Thus a core competence for a small independent mobile game producer could be its ability to version its software for many different mobile operating systems, with different hardware abilities and phone models.

Five Forces of Competition

Another way of understanding the success of a firm is by analysing the industry it must fit into. Thus a software firm is in competition with other firms; Apple, Microsoft, Google must survive in overlapping markets where their products partly compete for users (customers). The value of the software they produce is partly determined by how they compete in the market: thus their

competitive advantage. Porter[10] characterises five forces which determine the competitive structure of an industry (for example, developers of mobile operating systems).



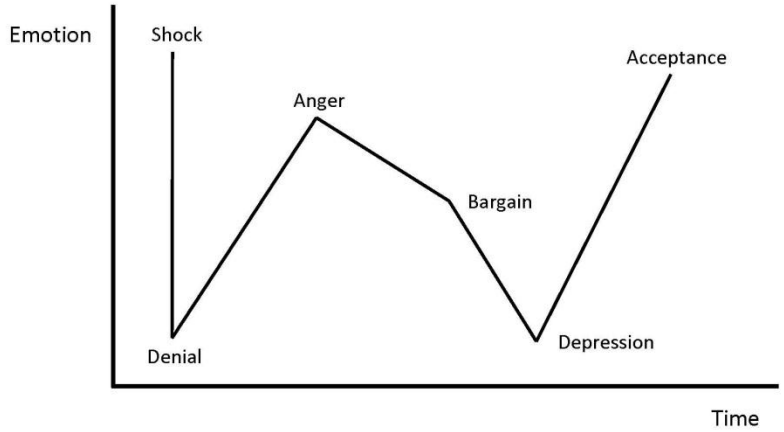
If we analyse the market for mobile operating systems we could look at:

- rivalry among existing firms – that is the competition for market share between Apple (IOS), Nokia (Symbian), Google (Android) and Blackberry (RIM BlackBerry OS) with their different strategies (proprietary/open source, own hardware/ manufacturer alliance.
- threat of new entries – Microsoft more or less fell out of this market, but made a re-entry attempt with Windows mobile 7.
- supplier power – not really an issue in the sense that Porter intended
- buyer power – smartphone buyers in the end determine the relative success of an operative system, but the decision is heavily influenced by the hardware package: iPhone or HTC or Blackberry.
- threat of substitute products – mobile operating systems could be challenged by (for example) cloud-based services run in a browser.

A software entrepreneur making this analysis might conclude that this is an extremely difficult market to break into.

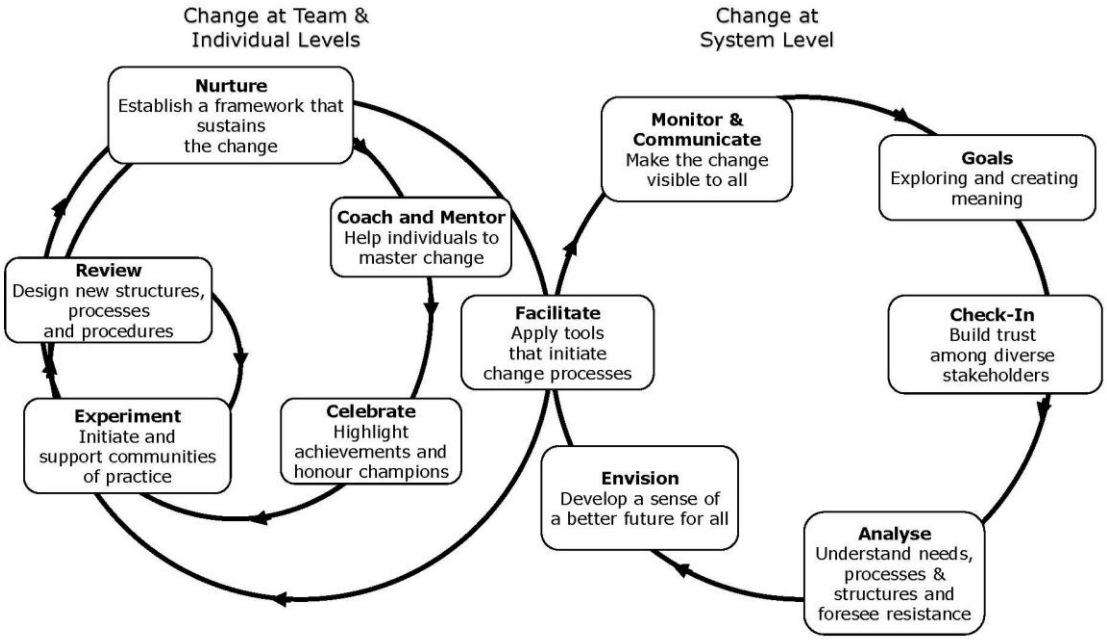
Theories of change and resistance

Theories of change will be important for understanding intrapreneurship because internal innovation in organisations is dependent on change. Organisations are not necessarily good at change; this is partly because they are made up of individuals who aren't good at change either. Their response to significant change can go something like this:



This could map, for example someone's response to a relationship breakup; however we should be careful with the idea that people leave their emotions behind when they go to work. Responses to change in organisations are characterised by the same kinds of feelings.

In larger organisations an intrapreneur needs to start at the local level, convincing his team and immediate line manager of the worth of his venture, and later work to change the whole organisational system.



Models of organizational change often start with an unfreezing stage – the notion that an organisation is relatively locked into its routines and practices, and something must be done to shake this up. They can finish with refreezing phases, were new practices are stabilised again. An example is Kotter's[11] eight stage organisational change model:

1. establish a sense of urgency (identify potential threats, and develop scenarios showing what could happen in the future, examine opportunities that should be, or could be, exploited, start honest discussions, and give dynamic and convincing reasons to get people talking and thinking, request support from customers, outside stakeholders and industry people to strengthen your argument)
2. create the guiding coalition (identify the true leaders in your organization, ask for an emotional commitment from these key people, work on team building within your change coalition, check your team for weak areas, and ensure that you have a good mix of people from different departments and different levels within your company)
3. develop a vision (determine the change values, develop a short summary that captures the future of your organization, create a strategy to execute that vision, ensure that your change coalition can describe the vision in five minutes or less, practice your "vision speech" often)
4. communicate the change vision (talk often about your change vision, openly and honestly address peoples' concerns and anxieties, apply your vision to all aspects of operations, tie everything back to the vision, lead by example)
5. empower broad-based action (identify, or hire, change leaders whose main roles are to deliver the change, look at your organizational structure, job descriptions, and performance and compensation systems to ensure they're in line with your vision, recognize and reward people for making change happen, identify people who are resisting the change, and help them see what's needed, take action to quickly remove barriers)
6. generate short-term wins (look for sure-fire projects that you can implement without help from any strong critics of the change, don't choose early targets that are expensive, reward the people who help you meet the targets)
7. consolidate gains and produce more change (after every win analyse what went right and what needs improving, set goals to continue building on momentum, practice continuous improvement, keep ideas fresh, bring in new change agents)
8. anchor new approaches in the culture (communicate progress, tell success, publicly recognize key coalition members).

We'll return to these theories as we discuss intrapreneurship: promoting new ventures from within the organisation.

Sources

BARNEY, J. B. (1996) The resource-based theory of the firm. *Organization Science,* 7**,** 469-469.

GRANT, R. M. (1996) Toward a knowledge-based theory of the firm. *Strategic Management Journal,* 17**,** 109-122.

KOTTER, J. P. (1996) *Leading change,* Boston, Harvard Business School Press.

PORTER, M. E. (1985) Competitive advantage: creating and sustaining superior performance, New York, Free Press.

PORTER, M. (2008) The five competitive forces that shape strategy. *Harvard Business Review,* 86**,** 78-86.

PRAHALAD, C. & HAMEL, G. (2003) The core competence of the corporation. *International Library of Critical Writings in Economics,* 163**,** 210-222.

The entrepreneurship landscape: themes and issues

> *'Entrepreneurship is the ability to create and build a vision from practically nothing .... It is the knack for sensing an opportunity where others see chaos, contradiction and confusion'*[12]

In this chapter we look at some of the themes and issues which are current in entrepreneurship research and will become relevant to our discussion of software entrepreneurship.

Entrepreneurship and economics

The roots of modern entrepreneurship (from the French: entreprendre, to undertake) theory lie in economic theory, with Schumpeter[13] as its founding father. He described the entrepreneur as an innovator engaging in a process of creative destruction by disrupting the circular flow of the market economics of production and consumption. Whereas these traditional market economics tend towards price equilibrium, the entrepreneur initiates new products and processes which replace the existing offerings, starting new firms which replace those that are no longer competitive. He characterised the entrepreneur as an innovator rather than a profit seeker, as an initiator and agent of change and a leader characterised by qualities of intellect, will, initiative, foresight and intuition. Schumpeter takes a form of macro-economic perspective in which the entrepreneur sees possibilities for new solutions unrecognised by others, and innovates to change the economic shape of the market. In an alternative view, Kirzner[14] offered a tactical, short-term and streetwise understanding of entrepreneurial behaviour. The entrepreneur discovers and exploits short-run price differentials as supply and demand move towards equilibrium. The entrepreneur is an alert opportunist, motivated by profit and constantly watchful profit opportunities arising from market disequilibrium, in which speed of movement and shrewd decision-making ability are essential. Entrepreneurship involves creative acts of discovery and learning, and he suggests that the entrepreneur outperforms others in the market through a superior ability to perceive opportunities and act on them. This is derived from the ability to learn faster than competitors.

Entrepreneurship and innovation

Innovation is often considered a vital aspect of entrepreneurship, particularly in high-tech ventures. Schumpeter, for example develops both themes in his account of macro-level economics. The ability of an IT firm to develop and sustain a market is often closely related to its ability to innovate[15] – to develop new products and services, and to learn new technologies. Software technologies develop at breakneck speed compared to some other fields. New software companies are often based on one, or few, products, often in niche markets dependent on highly specialised leading edge technologies – we will meet some examples later. We don't develop this theme too much because it is the subject of a companion volume (Software Innovation: eight work-style heuristics for creative software developers[15]).

The entrepreneurial type: underlying psychological traits or learned behaviour?

An entrepreneur is a person who creates organisations, or a person who recognises and acts to exploit an opportunity. This raises two questions

- are there are underlying personality traits which define a good entrepreneur?
- how much is entrepreneurship is a learned skill?

A recurring theme in entrepreneurship research has been the attempt to define a particular type of person who is an entrepreneur; as, for example, a psychological predisposition on the part of an individual to take a risk in the hope of gain. Need for achievement is claimed as a key motivator for entrepreneurial performance – dividing people into achievers and non-achievers. Entrepreneurs are sometimes understood to have a particular locus of control – the desire to control the world around their lives, rather than have their destinies determined by external factors. Rae[16] offers a list of entrepreneurial attributes which are typical of those considered in the literature:

- initiative
- strong persuasive powers
- moderate rather than high risk-taking ability
- flexibility
- creativity
- independence/autonomy
- problem-solving ability
- need for achievement
- imagination
- high belief in control of one's destiny
- leadership
- hard work

Personality and trait-based approaches to defining entrepreneurial behaviour have been criticised for their lack of consistency and inability to connect personality traits with actual performance. It is also questionable how far entrepreneurs exhibiting these characteristics differ from non-entrepreneurs - project leaders in software companies, for example. Nor is it clear how much such characteristics are hard-wired in entrepreneurs' personalities, and how much these behaviours are learned. More recent research has focused on how entrepreneurial skills are acquired, and Rae offers this summary of these trends. Research focuses on

- creative discovery learning that generates alertness to entrepreneurial opportunities
- the value of recent concrete experience (for example an IT project) related to context of use
- acquisition, storage and use of entrepreneurial knowledge as expert resource, for example, through reading these notes or attending a seminar
- key learning abilities within the small firm - how a start-up learns about its situation, environment and how to handle it
- the value of experience in entrepreneurial decision-making
- confidence and self-belief connecting learning resources with achievement
- the small firm as a dynamic entrepreneurial learning environment
- rational models of knowledge structures, cognition and decision making applied to stages of the entrepreneurial process
- dynamic learning processes with phases, processes and characteristics.

Entrepreneurship may have something to do with personality, but it's also clear that entrepreneurs come from all age ranges and backgrounds, with a wide variety of competences. Formal learning may play some part, but this is overshadowed by social learning: the ability to learn rapidly from ones situation, experience and context.

Entrepreneurial opportunities: discovered or created?

The exploration and exploitation of opportunity lies at the heart of entrepreneurial activity. Do entrepreneurs discover and recognise opportunities which already exist, or do they create and enact new opportunities? Are opportunities the result of what we see (a potential niche in a market), or what we do (build an innovative application)? In the first case an entrepreneur should excel at understanding the markets in which they operate, looking for weaknesses in their competitors' products and services, recognising unfulfilled customer needs and planning business activities that realise the discovered opportunities. In the second case they should operate with vision and drive to develop products and services they believe in and work to create the necessary markets – promoting their activities and developing their customer base. This discussion is extended later in the paradigm discussion. Rae concludes that

> 'short-term opportunities do exist and await discovery by the alert entrepreneur. However the circumstances that can give rise to new opportunities may be recognised by people with the imagination, experience and judgement to do so, and that they can create and enact the opportunities which they assess as worthwhile. The vital skill lies in recognising the future opportunities with the greatest long-term potential which the entrepreneur can gather the resources to exploit.'

However, Sarasvathy's research into effectuation[17] is heavily based on the premise that we create our own opportunities and decide our own futures.

High-tech entrepreneurship

The software market is a high-technology market characterized by high levels of market and technological uncertainty. Convergence of the software, telecommunications and media industries creates many new opportunities for software firms. There are new types of firms, technological bases, products and services, and new value propositions for different customer segments[18].

Some special characteristics of high tech firms affecting entrepreneurship are:

- low cost of entry writing software
- virtually free distribution through the internet
- international through English language
- often not necessary to live in a particular place (e.g. close to customers because of high delivery costs)

Entrepreneurship and business planning

In some accounts of entrepreneurship (not this one) the central activity is business planning[19, 20]. The idea is that the preparation of a new venture, through careful environmental scanning (for

example analysing the industry environment with Porter's five forces model), opportunity analysis, marketing research, and financial planning (budgeting, break-even analysis) is the keystone of building a new business. The result of such activities is the business plan, which is the cornerstone for future activities. There's nothing wrong with this approach (which reflects business preoccupations which are also important for software entrepreneurs), but they're not the things that usually motivate IT professionals and engineers. We'll discuss this style of entrepreneurship thinking further under the heading of Analyse, Design, Enact. There are many practically-oriented guides for doing this easily available on the net, and if you need to do this you should consult an appropriate reference from the end of this section.

Entrepreneurship: myths and fears

In this section we tackle some common assumptions about entrepreneurship which are known to be mistaken, and some widespread fears which are unnecessary.

Entrepreneurs are visionaries

In the IT world, many associate leadership with some extraordinary figures with visionary ideas: Bill Gates at Microsoft, Steve Jobs at Apple, Pierre Omidyar who founded eBay, Mark Zuckerberg with Facebook, Jeff Bezos at Amazon, Sergey Brin and Larry Page at Google. All these companies were once start-ups, and their founders at that time software entrepreneurs. Does this mean you have to be a visionary to be a software entrepreneur? Not really. Take a look at the structure of the software industry, the eCommerce landscape and the developing mobile services and applications market. For each of these hugely successful companies there are a thousand smaller companies, some highly innovative, some relatively traditional, and each of these was also a start-up with a software entrepreneur. If you have a good idea and the drive to start a software company you simply don't know whether you will be a great visionary, a successful director of a medium sized firm, the leader of a small highly-focused and innovative firm, a serial entrepreneur (someone who founds several companies and moves on) or someone with a company that goes bust. You'll find out whether you are a visionary later!

Entrepreneurs are risk takers

If you are going to be an entrepreneur you need to mortgage your house to raise money, you will owe the bank a small fortune, you will work for many years without any real salary and if it goes wrong you will lose your house and family, be declared bankrupt and be unemployable for the rest of your life. These are scenarios that can become real for a small number of entrepreneurs, but they do not match any general empirical picture of how entrepreneurs work and what the outcomes of their lives are. The majority of successful entrepreneurs end up running small and medium sized businesses with a comfortable level of risk, have salaries comparable with wage earners with similar responsibilities, and eventually choose to use their privileged status as owners to generate some leisure time. Comparative studies of bankers and entrepreneurs show that bankers work with significantly higher risks, offsetting them with insurance[21]. Entrepreneurs usually choose to work with comfortable levels of risk, expanding their portfolio of activities to generate opportunities which offset their risk profiles. Risk and uncertainty are a part of work life (wage earners can be fired, and their projects can bomb) and need to be understood and managed, but the entrepreneur is not in a situation of unusually high risk unless they choose themselves to be there.

Entrepreneurs are not like us

This myth expresses the idea that entrepreneurs have a particular profile, which no one can really define, but we are quite sure is different from our own. Actually, as discussed above, this profile can't be determined apart from in general terms common to many with some degree of success in life, and it appears that most of the necessary skills are often learned. It might be the case that some people are not suited to be entrepreneurs, but if you have an idea you shouldn't let this myth put you off. As a software entrepreneur you will statistically have a somewhat better chance of success after you have some professional experience, but before your life is really settled in a particular pattern in middle-age. The median age for starting the first business for entrepreneurs who graduated from MIT is thirty[22]. You shouldn't let this put you off either, the differences in outcomes are not that huge.

You need a brilliant idea

A new internet search algorithm (Google), a net-based book retailer (Amazon), a new way of managing social relationships on the net (Facebook), auctions on the net (eBay), a new PC interface based on the metaphor of the desktop (Apple); you need a brilliant idea to start a successful company. What about Twitter? The idea of a short status update was in reality just a limitation imposed by the prevailing short message service standards and protocols in the mobile industry. It was at the time technically hard to organize longer messages – but users could see a point in it: the immediacy of being able to report what they were doing many times a day, wherever they were. The success of the company is dependent on being able to recognize what their users find important, not on the original idea. Everyone has ideas; hardly any of them are implemented with enough drive and conviction to find a market. A good idea is first an idea that has been made a reality; then a reality that can generate value. Many software entrepreneurs start instead with customers – if you have someone that asks you to implement their web-site or build them a mobile service, you have no idea of your own, but you have a business possibility. The ideas can also come later, or not at all. You can also work with the opportunity and doability[21]. Is my venture

- doable (is it feasible, can it be programmed, will somebody pay for it)?
- worth doing (can it generate an income I can be satisfied with)?

and

- can I do it (do I have the necessary skills and resources?)
- do I want to do it (does it excite and challenge me, can I see a way out if it doesn't go well)?

You need a lot of money

Zaplet was a Silicon Valley start-up developing dynamic, updateable messages and applications through email. In 2000 they had 27 product managers and 30 developers and US$90m in venture capital. GO corporation survived six years and went through US$75m of venture capital developing a pen operating system. You've never heard of these products or companies because they went bust. If they can't succeed with all that capital, how can you with those meagre resources you can muster? Here's the other side of the story. Dell Corporation was founded in 1994: starting capital US$1,000. A 2002 survey of the 500 fastest growing companies in 2002 showed that 14% started with less than US$20,000. 98% of new US businesses start with no venture capital or angel funding[21]. What do you need to begin writing and selling a smartphone game apps? Some development software and an emulator (all available free), a laptop to write it on and a smartphone to see if it works (you've

got these already) and a developer licence (iOS developer license: $99). Oh – and your creativity and engineer skills. Still think you need a lot of money?

Sources

DOLLINGER, M. J. (1999) Entrepreneurship: strategies and resources, Upper Saddle River, New Jersey, Prentice hall.

KAPLAN, J. (1994) Start Up, London, Warner.

KIRZNER, I. M. (1978) *Competition and Entrepreneurship,* Chicago, University of Chicago Press.

KURATKO, D. F. & HODGETTS, R. M. (1995) *Entrepreneurship: A contemporary approach*, Dryden Press.

KURATKO, D. & HODGETTS, R. (2004) Entrepreneurship: Theory, Process and Practice, Mason, Ohio, Thomson.

RAE, D. (2007) Entrepreneurship: from Opportunity to Action, New York, Palgrave.

READ, S., SARASVATHY, S., DREW, N., WILTBANK, R. & OHLSSON, A. (2011) *Effectual Entrepreneurship,* London, Routledge.

ROSE, J. (2010) *Software Innovation: eight work-style heuristics for creative software developers,* Aalborg, Software Innovation, Dept. of Computer Science, Aalborg University.

SHUMPETER, J. A. (1934) The theory of economic development. *Cambridge, MA: Harvard*

Two paradigms for software entrepreneurship

Why paradigms?

There are three major reasons for adapting a paradigmatic way of thinking about software entrepreneurship:

1. it helps us structure a complex field and delineate it in a relatively approachable way - with the risk, of course, of over simplifying
2. it helps us to focus on Sarasvathy's effectuation theory, which has some advantages for young software and IT professionals (it's more likely to reflect the ways you do things) without losing sight of other important parts of the field
3. we can easily relate it to another familiar discussion with which we are familiar and which we understand quite well - the comparison of agile and traditional development approaches.

Two paradigms from entrepreneurship thinking

We'll develop two paradigms for software entrepreneurship, based on the work of Saras D. Sarasvathy [17]. She works in the general field of entrepreneurship, so we also need to adapt her ideas so they have more relevance for software professionals. Her work is interesting because it provides a very direct challenge to classical ways of thinking about the field. We'll call the two paradigms 'Analyse, Design, Enact' (ADE) and 'Consider, Do, Adjust' (CDA). ADE reflects the classical way that entrepreneurship has been researched and taught in business schools as represented by many of the major textbooks in the area. In common with traditional ways of thinking about software development, the focus here is on analysis and design - an intellectual process of finding out about a new venture in an economic and technical environment, and conceptualising it as fully as possible, before setting it in motion. In CDA the focus is on action: not that you should do things without sensible consideration, but that action should be taken, a venture set in motion on the basis of what is currently possible and then adapted according to developing circumstances - an agile way of thinking. In the following short analysis we'll look at ADE as represented by Kuratko[19, 20] and Dollinger[7] (leading textbooks), and then follow Sarasvathy's arguments as she outlines the assumptions in this thinking and contrasts it with her own.

ADE: Analyse, Design, Enact

ADE emphasises an intellectual process which is analytical in respect to the current state of the world and how it can be understood. Dollinger breaks the process down into scanning, monitoring, forecasting and assessing, and expects a considerable investment in learning about markets, economies, technologies and business conditions. The issues that should be analysed include:

- Political and governmental issues
    - Stakeholder analysis
    - Global and international issues
        - Trade barriers and tariffs

        - Trade agreements
        - Political risk
    - National issues
        - Taxation
        - Regulation
        - Antitrust legislation
        - Patent protection
        - Government support
        - Legal structures
    - Regional issues
        - Licensing
        - Securities and incorporation laws
        - Incentives
- Macro-economy issues
    - Structural industry change
    - Cyclical change (e.g. financial situation)
- Technology issues
    - Invention
    - Process innovation
    - Trends
- Social demographics issues
    - Demographics
    - Social trends and values
- Ecological issues
    - Sustainable development
- Market issues
    - Segments
    - Consumer behaviour
- Industry Competition
    - Similar services and products
- Opportunity

Analysis is safest where there is a strong theoretical background (this adds a history of practice in what to look at and how to interpret the results) so many models are presented, mainly from standard business literatures (for example the work of Michael Porter). It's usually necessary to collect empirical data to make the analysis. We looked earlier at the operating system industry with the help of Porter's five forces model and could easily conclude that this was a difficult market to break into with the scale of resources at our command.

With an accurate picture of the business conditions, it's possible to *design* a response - a venture which will fit into and match those conditions and therefore be successful. Typical design activities are:

- Business model
- Marketing strategy
- Financial planning

- Organizational design (see for example Mintzberg)
- Legal structure
- Work-practice design
- Product design
- Growth strategy

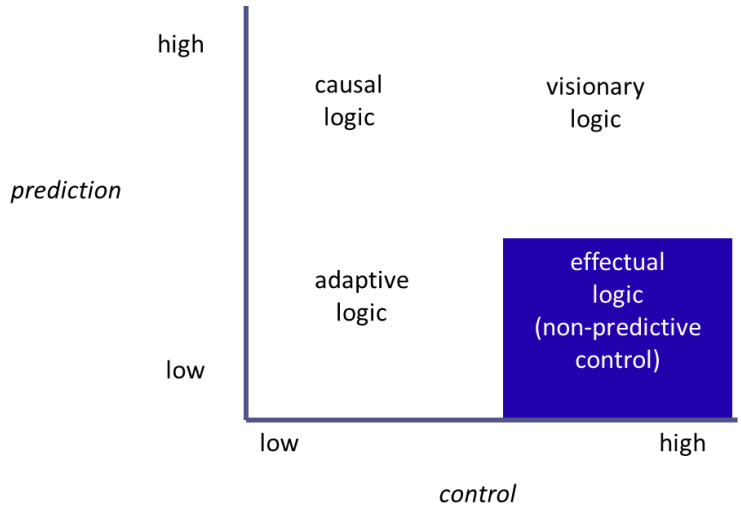
These can also have a strong theoretical component, for example a business model can be based on a documented category of business models, such as those described by Rappa. These types of normative models help rationalise the design process, again using historical examples which are known to be successful. Design is complemented by planning - so the medium term future of the venture can be anticipated and the necessary resources obtained, goals and expectations set. The plan provides a management tool - a way of understanding whether the expected targets are being met, and what corrective action to take if they are not. The result of the design exercise is a business plan - a document summarizing the designs and plans. You could think of this as a specification for the new venture.

business plan

1. Executive summary
2. Background and purpose
3. Objectives
4. Market analysis
5. Development and production
6. Marketing
7. Financial plans
8. Organization and management
9. Ownership
10. Critical risks and contingencies
11. Scheduling and milestones
12. Conclusions

The business plan is the basis for *enacting* the venture: sponsors and investors must be found, venture capital raised, a legal structure established, premises found and the necessary equipment bought, staff recruited. The business plan is the primary vehicle for attracting capital. The business can then be managed according to the plan; once it starts to generate income the investors can be repaid.

Sarasvathy argues that way of thinking represents a causal logic: to the extent we can predict the future, we can control it. This is useful when the future is uncertain, but knowable, and goals are clear, but ways to achieve them are not. Causal logic assumes that the external environment is reasonably well-structured, but largely outside our control.

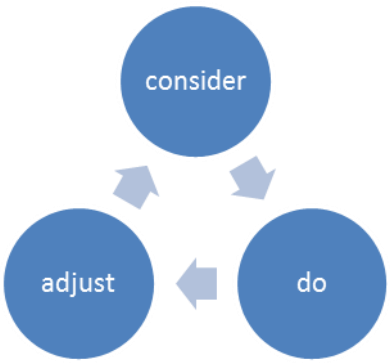


The vertical axis assesses how predictable the future is; essentially whether it can be modelled by using contemporary and historical data and a reliable theoretical tool. The horizontal axis assesses how much the future is under our control: whether we have the potential to determine it to a larger or smaller degree. The matrix yields four logics of entrepreneurship.

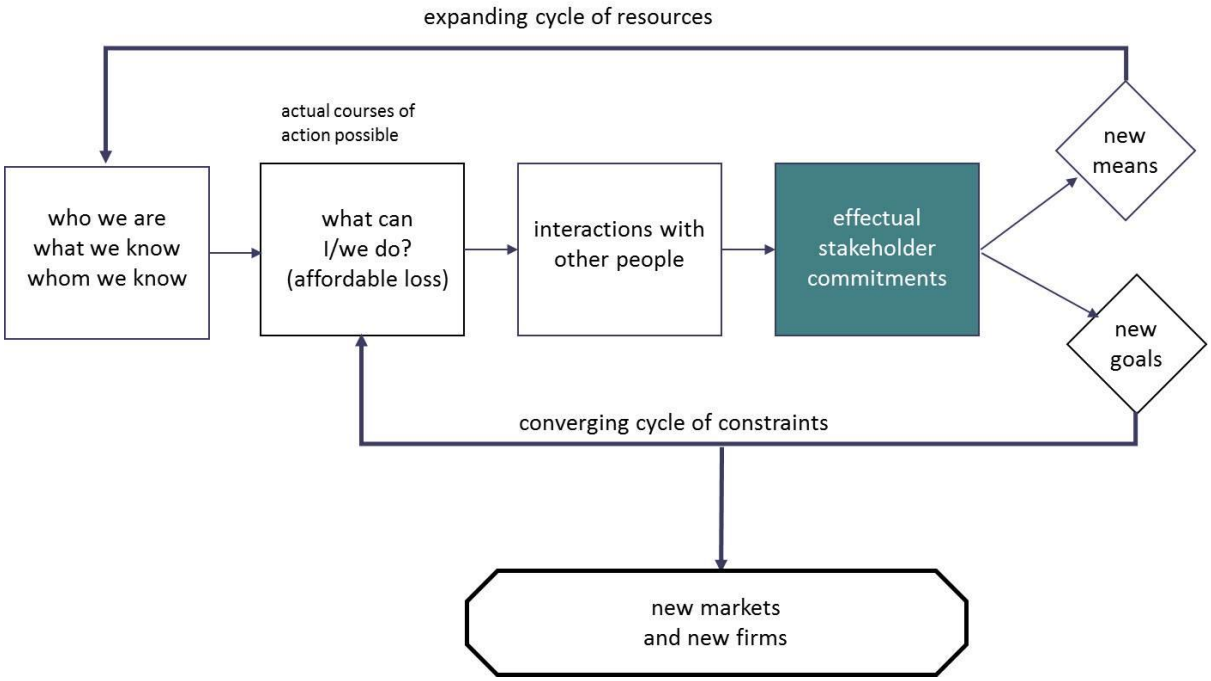
- the causal logic behind the ADE paradigm (we must fit into a market and industry which can be analysed, but not controlled)
- visionary logic - we can both predict the future and have a high degree of control over it (Steve Jobs introducing a major new product at Apple; the market gap and technology trends are known, the market is heavily influenced by Apple's brand and marketing)
- adaptive logic – low predictability, low control (a start-up with one large industrial customer and an untried technology)
- effectual logic – non-predictive control (see below)

CDA: Consider, Do, Adjust

In her theory of effectuation (the act of implementing, providing a practical means for accomplishing something; carrying into effect) she argues for the effectual logic over the causal, showing that it is more common in (a sample of) successful entrepreneurs. An effectual logic assumes that the future is created by the actions of people, and cannot necessarily be predicted. Instead the entrepreneur can take control of their resources and work to influence the future. If there is no product, then the product can be created, if there is no market, then the market can be created. Effectual logic suggests that to the extent we can work with things within our control, we don't need to predict the future, and is useful when the future is not only uncertain, but also unknowable, goals are ambiguous (but means are clear and limited), and the environment is unstructured and subject to shaping by human action. Here it makes less sense to invest in large scale analysis and design, and more sense to act. Actions are not without reflection, but they take centre stage, and future actions are in response to those types of future that we were able to create - an iterative cycle: Consider, Do, Adapt.



Here is Sarasvathy's account of the effectuation cycle:



The starting place is with the entrepreneur's knowledge and abilities; with their possibilities for action now (e.g. we have the resources to build an android leisure game, but not to port it to other platforms). It's essentially a networking theory of entrepreneurship, emphasising interactions with, and commitments from others (e.g. we know someone who has some experience working with iPhone games, maybe they'd come on board). New partnerships create further means (resources) and new goals and constraints (now we're a multi-platform leisure game developer, but we still don't have the resources to develop large-scale games). She also provides some effectual principles:

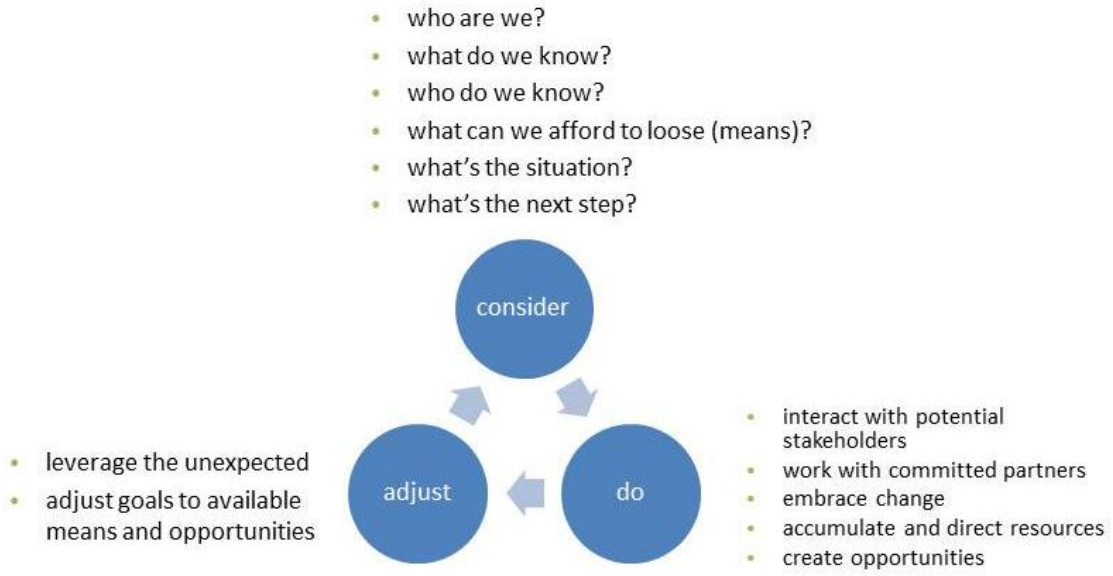
- The bird-in-hand principle: start with who you are, what you know, and whom you know (rather than with pre-set goals expressed in a five-year business plan)
- The affordable loss principle: invest what you can afford to lose – in the extreme case nothing (rather than borrow a lot of money and calculate the expected return in five years)
- The crazy quilt principle: build a network of self-selected stakeholders (not market analysis to find out where you fit)

- The lemonade principle - when life sends you lemons (unexpected surprises) make lemonade: embrace and leverage surprises (don't avoid them and try to stick to the business plan)
- The pilot-in-the-plane principle: the future comes from what people do (not inevitable trends in technologies, markets and industries)

Here are the differences summarized:

causal logic		effectual logic	
goals first	design the venture upfront, specifying what you want to do and where you want to be	*means first*	let the venture emerge out of what you are able to do with the resources you have
maximize return	borrow money and use the venture to increase its value and leave a profit	*control investment*	invest what you are comfortable with and see what you can achieve with it
avoid the unexpected	stick to the plan and overcome obstacles	*leverage the unexpected*	change the direction of the venture to respond to the opportunities you meet
think competition and market	fit the venture into its niche in a jigsaw of market and industry conditions	*think pre-commitment*	make partnerships and alliances with others who are prepared to commit to you
predict and control	develop a roadmap and act according to it	*control don't predict*	use the available resources to develop achievable results

If we organise Sarasvathy's thinking as the Consider, Do, Adjust paradigm we get something like this:



We can express the difference between these two paradigms relatively clearly with the help of Sarasvathy's work, but considerable challenges remain before they can really be meaningful to software professionals. The first challenge is that they are too generalised - most software professionals would have good reason to believe that software ventures have special characteristics

(high tech, high growth, intellectual property, knowledge intensive and so on) which mean that they need a specialised form of consideration. The second challenge is that most of the entrepreneurship literature is written for business people, not for engineers. Engineers build things, and software professionals are no exception, so we need to find ways to focus on interesting software ideas and how they are developed. Business is only the vehicle which enables engineers to write interesting software. We could also add the challenge that software is a strongly innovative industry, and that many of the forms of entrepreneurship that we are interested in involve an innovative ('interesting') software product. A final challenge could be to do with the structure of the software industry; we need to develop entrepreneurship models for consultant- and product-oriented (and other types of) firms.

In the next section, we'll adapt these two paradigms to our theme: promoting software and IT ventures.

Analyse, Design, Enact in the software context

A more software-oriented sketch of the ADE paradigm might look like this. Imagine we want to set up a company to provide time-stamped databases to industry. The things we might like to *analyse* could include:

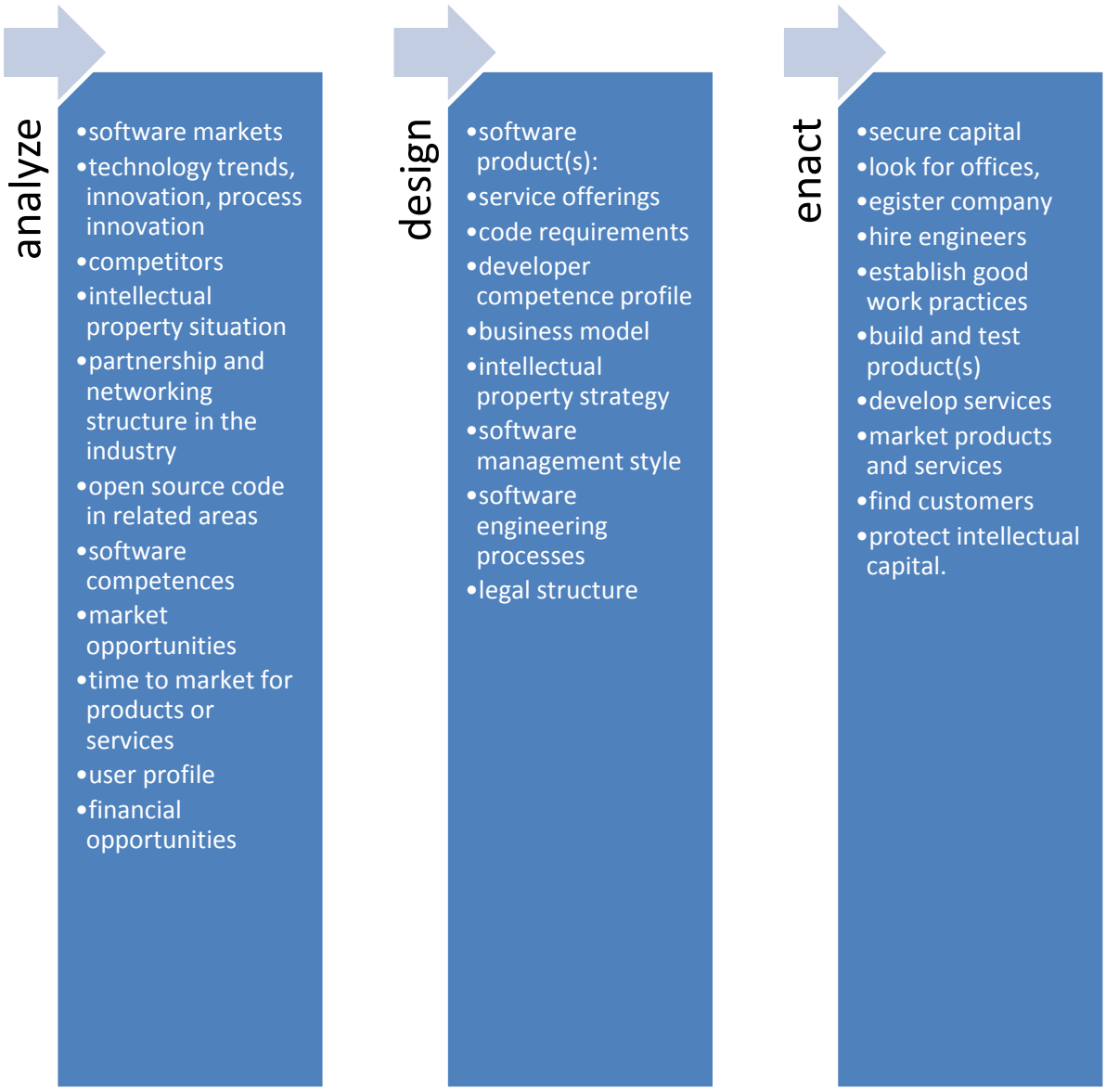
- software markets
- technology trends, innovation, software process innovation
- competitors in chosen software market with similar products and services
- intellectual property situation (patents and copyrights in similar areas)
- partnership and networking structure in the industry
- open source code in related areas
- our own software competences and any gaps
- market opportunities
- time to market for products or services (estimation of developer hours)
- user profiles (segments, behaviour, demographics, social trends and values)
- the financial opportunities available to us

Then we need to make a business plan in which we *design* our company; we should specify (for example):

- our software product(s): low-tech and code prototypes, specifications
- our service offerings (implementation, customization, consultancy)
- own code requirements (what we need to have a functioning product that can be implemented at a customer)
- developer competence profile (which programming technologies we can work with, which types of platform and applications we have expertise in) and how we need to build it
- the business model for the venture (how we will earn our revenues)

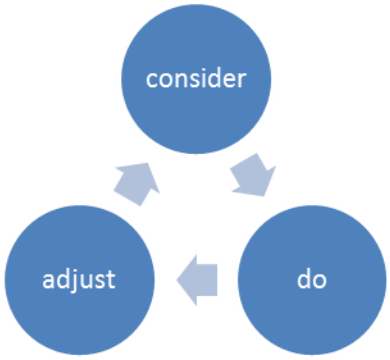
- our intellectual property strategy (closed, open, mixed)
- software management style (what kind of developer culture do we want to encourage at our company)
- software engineering processes and models (traditional, agile, test and quality approach)
- the legal structure for the company

With our business plan in hand we can go into the *enact* stage: we should secure capital and look for offices, register our company, hire some engineers and establish good work practices, build and test our product(s) and develop the services we will offer. Now we need to market our products and services in order to find some customers and make sure that we protect intellectual capital.



If we look at the assumptions behind this way of thinking we expect that the software venture will compete in an existing market (not create a completely new market). We assume that market and technology developments are foreseeable on the basis of historical evidence (we analyse the past to predict the future) and that opportunities in these markets can be discovered and exploited. We assume that a software venture can designed in such a way as to exploit the discovered opportunity and that our software competences should be developed in line with the design. We assume that investors will invest on the basis of convincing analysis and design work, and, rather fundamentally, that the future can be planned for (it is predictable and knowable). We assume we need venture or loan capital, that the plan is a safe basis for setting things in action (enacting), and that managing the new venture is primarily controlling that the plan is enacted precisely.

Consider, Do, Adjust in the software context



A more software oriented version of CDE could look like this:

We should *consider*: which specialised software competences and computer science (or IT) research interests do we have, what existing code and prototypes do we have lying around? How much of our time can we use for developing? What programming and project management skills do we have that customers might want to hire? What services can we offer to our customers? How can we best improve or differentiate our products with the development time we have available? Can we improve our engineering practice? How can we involve others with complementary software competences or code and applications that we need, and will they commit to our projects? Who do we know that is a potential customer and how can we work with them to improve our products?

We should act (*do*) to: build software and develop services, actively seek and interact with (potential) customers (using their skills and domain knowledge to develop better applications). We should actively seek and interact with (potential) partners and co-operators to develop our mutual programming and development competences and expand our sellable code-base. We should expand and diversify revenue generation (more products and services) and employ financial bootstrapping principles to generate more resources for our projects. We should share our knowledge and learn from those who are prepared to do the same, and leverage the possibilities of open source code resources.

We should constantly *adjust* by: modifying our products and services according to customer feedback, by identifying new products and services through customer and partner interaction, by

dumping dead-end projects and opening new projects as resources allow, and by improving internal engineering practice as a result of our experience.

In this model we assume that the future software market is unknowable, but created by our current developments, that market for a software product (especially innovative products) can be created if it does not exist, and that our environment is too complex to be perfectly analysed. We further assume that action (developing software) with imperfect knowledge is better than planning, as long as it is constantly adjusted to fit external customer realities, that developing software is more useful than business planning, as long as business development is actively considered, and that interactions with partners and customers are the key to building good software and developing business potential. We think knowledge sharing promotes mutual learning and this is important because software is primarily about knowledge, that lean is good, and that managing a young software company means developing good (appropriate) engineering practice.

Case study: Pierre Omidyar on starting eBay

Almost every industry analyst and business reporter I talk to observes that eBay's strength is that its system is self-sustaining -- able to adapt to user needs, without any heavy intervention from a central authority of some sort. So people often say to me – 'when you built the system, you must have known that making it self-sustainable was the only way eBay could grow to serve 40 million users a day.' Well… nope. I made the system self-sustaining for one reason: Back when I launched eBay on Labour Day 1995, eBay wasn't my business - it was my hobby. I had to build a system that was self-sustaining because I had a real job to go to every morning. I was working as a software engineer from 10 to 7, and I wanted to have a life on the weekends. So I built a system that could keep working - catching complaints and capturing feedback -- even when Pam and I were out mountain-biking, and the only one home was our cat.

If I had had a blank check from a big VC, and a big staff running around - things might have gone much worse. I would have probably put together a very complex, elaborate system - something that justified all the investment. But because I had to operate on a tight budget - tight in terms of money and tight in terms of time - necessity focused me on simplicity: so I built a system simple enough to sustain itself. By building a simple system, with just a few guiding principles, eBay was open to organic growth - it could achieve a certain degree of self-organization. So I guess what I'm trying to tell you is: whatever future you're building… don't try to program everything. 5 Year Plans never worked for the Soviet Union - in fact, if anything, central planning contributed to its fall. Chances are, central planning won't work any better for any of us.

Build a platform - prepare for the unexpected... …and you'll know you're successful when the platform you've built serves you in unexpected ways. That's certainly true of the lessons I've learned in the process of building eBay. Because in the deepest sense, eBay wasn't a hobby. And it wasn't a business. It was - and is - a community: an organic, evolving, self-organizing web of individual relationships, formed around shared interests

Paradigm comparison

The two paradigms should be understood as the idealised theoretical endpoints of a continuous spectrum, which is useful for understanding and categorising the literature. In reality most software organisations will use elements of both. A rigorous planning exercise does not exclude flexibility, rapid change and iterative learning in implementation, and a bootstrapped iterative growth model does not exclude some elements of rigorous analysis, for example a thorough market analysis for a new product. You shouldn't expect to find real empirical examples which are perfect examples of one paradigm or the other. In some related areas the 80/20 rule applies - suggesting that you should be 80% focused on one approach, and that 'stuck in the middle' positions do not work, but there is no empirical justification for this at the present time in this field. The two paradigms are compared in various dimensions below[23].

	analyse, design, enact	consider, do, adjust
theoretical inspiration:	standard entrepreneurship literature	Sarasvathy: theory of effectuation
software engineering inspiration:	traditional development	agile development
description:	promote a software venture by understanding the technical and business environment, designing a logical response and setting it in motion	promote a software venture by taking iterative and incremental steps forward on the basis of what is achievable now
process:	sequential	iterative
understanding of future:	predictable, non-controllable – adjust to the forces of the market	unpredictable, controllable – create (a small part of) the future
attitude to market:	analyse the market and choose a position or niche (find opportunity)	avoid or outsmart obvious competitors and create a new market (make opportunity)
attitude to technology development:	understand technology trajectories and fit software projects into likely developments	develop the areas of technology expertise you excel in
role of business planning:	conceptualise the venture as completely as possible before starting	take initial business decisions based on what you can afford and look out for what you need to do next
software development style:	decide software engineering practice up-front, likely to emphasise the traditional	improvise engineering practice and decide according to circumstance, lean practice, agility
attitude to change:	avoid change as far as possible - stick to plan to achieve goals	embrace change as opportunity and act upon new situations and possibilities
funding approach:	attract capital and investors before start up - fund to maximize return	invest what you can afford without considering the possible return and bootstrap
approach to others working in the same areas:	avoid competition - consider competitors a threat to market position	collaborate with those who demonstrate commitment and network with potential stakeholders
approach to intellectual property:	protect IP through copyrights and patents to deter competition, improve market share	build networks of ideas, sharing, various business models including open source, mixed open and closed code access
partnering and networking	control collaboration to avoid losing intellectual property	build a network of committed stakeholders, collaborators and partners
time to market	long on the basis of venture capital	short to maximise returns

Sources

CHESBROUGH, H. (2007) Open business models: How to thrive in the new innovation landscape, Harvard Business Press.

CUSUMANO, M. (2004) The business of software: What every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad, Free Press.

DOLLINGER, M. J. (1999) Entrepreneurship: strategies and resources, Upper Saddle River, New Jersey, Prentice hall.

KAPLAN, J. (1994) Start Up, London, Warner.

KURATKO, D. F. & HODGETTS, R. M. (1995) *Entrepreneurship: A contemporary approach*, Dryden Press.

KURATKO, D. & HODGETTS, R. (2004) Entrepreneurship: Theory, Process and Practice, Mason, Ohio, Thomson.

SARASVATHY, S. (2001) Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. Academy of management Review, 26, 243-263.

SARASVATHY, S. (2003) Entrepreneurship as a science of the artificial. Journal of Economic Psychology, 24, 203-220.

SARASVATHY, S. (2004) Making it happen: Beyond theories of the firm to theories of firm design. Entrepreneurship Theory and Practice, 28, 519-531.

SARASVATHY, S. (2008) Effectuation: Elements of entrepreneurial expertise, Edward Elgar Publishing.

SARASVATHY, S. & DEW, N. (2005) New market creation through transformation. Journal of Evolutionary Economics, 15, 533-565.

WILTBANK, R., DEW, N., READ, S. & SARASVATHY, S. (2006) What to do next? The case for non predictive strategy. Strategic Management Journal, 27, 981-998.

The software start-up

Much of the focus of the entrepreneurship literature is on starting a new business, so in this section we look at software start-ups, companies where the primary value creation is achieved through writing software.

Giarratana[24] mentions three types of entrepreneurs:

- the *innovator* creates new products (our software is ground breaking and has not been seen before)
- the *arbitrageur* exploits market inefficiencies (there's only one company that provides ecommerce solutions for local businesses and they're too expensive for the region's small firms - we can provide a personalized, low-cost alternative)
- the c*oordinator* introduces an alternative use of resources (we will provide weather presentation software based on open source components where much of the development effort is free which will compete with the dominant proprietary software providers such as Metra).

In each case the entrepreneur will have to understand something of the business positioning of the new company. Cusumano[1] suggests that there are some basic questions that need to be answered:

- do you want to be mainly a *products company* or a *services* (consultancy) *company?*
- do you want to sell to *individuals* or *enterprises,* or to *mass* or *niche* markets?
- how *horizontal* (broad) or *vertical* (specialized) is your product or service?
- can you generate a *recurring revenue stream* to endure in good times and bad?
- will you target mainstream customers, or do you have a plan to avoid *"the chasm"* (the gap between early adopters - the rather few customers with specialised interests who buy technically innovative products, and the early majority (the many customers who want to buy a product that is technically sophisticated, but also established and popular)?
- do you hope to be a *leader* (a company that is an innovator)*, follower* (one that bases its products on established technologies)*,* or c*omplementor* (a company that provides additional software and services to an established platform)?
- what kind of *character* (culture, image, brand, working environment) do you want your company to have?

The answers to these questions may (in an Analyse, Design, Enact perspective), serve to help build strategies for service and market development[25]. Software firms can be customer-oriented or competitor oriented[26]. Gathering information on customers, meeting their needs and creating value for them are essential ingredients for a customer oriented business, which synchronize well with user-centred forms of software development. Competitor orientation can be understood as company understandings of strengths, weaknesses, capabilities and strategies of key potential competitors, including an understanding of the applications they develop and the consultancy

services they offer. Mueller[25] finds that competitor orientation has a positive correlation with technology success for software start-ups, but that customer-orientation should either be taken very seriously or left out altogether . Start-ups may have both technology and business strategies[25]. Generic technology strategies include:

- leader (building image, innovation skills, coupling technological advances with user needs)
- follower (feature optimization for established product types, design skills)
- exploiter (quality maximisation, modification and re-engineering skills)
- extender (price minimization, vendor skills)

These can be matched with business strategies including

- cost leadership (being able to build software more cheaply than others)
- differentiation (having software products or services which are recognizable unique or different from standard offerings)
- focus or niche (competing in a specialised area with few competitors)
- speed to market (the ability to innovate and bring applications to the market faster than competitors

and the development of core competences such as technical expertise, design and innovation expertise, software engineering expertise, and market understanding.

Malaysia's government policy for entrepreneurship development, especially in the IT industry, offers financial and other help for funding new projects, and supporting corporate R+D. However, the most frequent sources of finance are personal funds of entrepreneurs and venture capital, and less use is made of loans from the government and commercial banks. Most employees are technically educated, and a third of the software engineers hold a degree in computer science from developed countries, such as USA, Australia, New Zealand, Canada and UK. With the help of these foreign-trained professionals, the firms have acquired the latest software technology from abroad. Sharing typical characteristics with entrepreneurial firms in other industries, technical and commercial knowledge is located at the top and concentrated in specialized functions. Management style is highly dynamic and flexible, and the organizations are structured as open systems: fluid and result-oriented. Management highly values contributions from individual employees, and the speed of deciding whether or not to adopt a new idea is fast. In most firms top management continuously scans the business environment for the latest software and market trends. The start-ups' main competitors are domestic companies, but competition also comes increasingly from international firms. Most software entrepreneurs feel the need for going international as somewhat urgent, though the Malaysian software market still provides large potential opportunities. Aiming at these market opportunities, some entrepreneurs explicitly include in-house innovation into their strategic plans. The entrepreneurs report serious problems in finding technically educated people and have to bear high costs for qualified personnel. They mainly design customized software to meet the demand of local customers. Some firms also do business with big international software companies, either as subcontractors or as distributors. It should be noted that most entrepreneurs started their businesses as vendors, gathering experience and technological know-how, and later began designing their own software. Most adopt differentiation as their product strategy, or work in a niche market; in customized software design they focus on cost leadership. Malaysia's software market has been

strongly influenced by foreign companies, and the start-ups have to compete with high quality software products. If a firm is unable to differentiate its software in terms of product quality, type and customer segments, it cannot increase sales. Most ventures adopt a technology exploiter strategy. The entrepreneurs place highest importance on know-how technology, i.e. human skills and expertise. Entrepreneurial software firms in Malaysia are competitive in designing high quality customized software but do not have sufficient resources and capabilities for developing new software on their own. They face problems with innovation[25].

These considerations hint at different ways of generating value out of software applications, so we investigate this next.

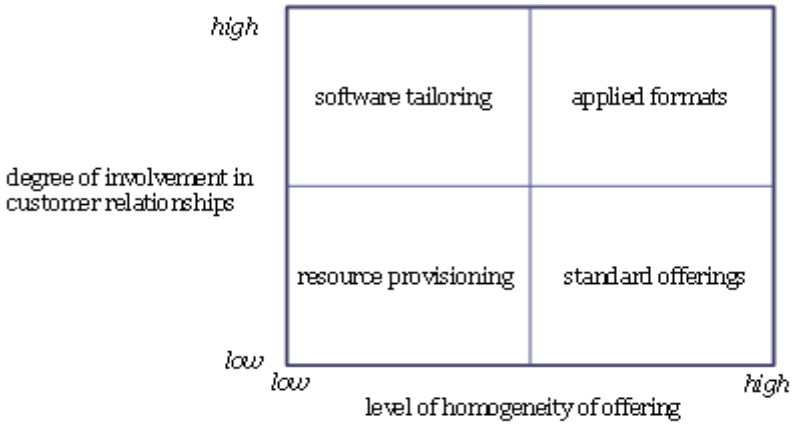
Software firm business models

If you're starting a software business, what type of business will it be? What will it offer in terms of products and services and how will it generate value? The most common distinction is between companies primarily offering products, and those primarily offering services[1]. Product companies (Microsoft, Adobe, Apple) build packaged proprietary software and sell it to customers on the open market. Service (consultancy) companies (PricewaterhouseCoopers, Accenture, Cap Gemini Ernst and Young) contract with customers to develop software tailored to their needs. Many mature companies develop effective hybrid strategies, but start-ups, according to Cusumano[1] need to choose and continue to choose in their early growth phase.

According to Rajala [27, 28], a software business model consists of four elements (a somewhat narrow characterisation compared to other writers on the subject):

- a product strategy (what kind of software and applications do we write)
- a revenue logic (how do we earn money out of them)
- a distribution model (how do we get them to our customers)
- a service and implementation model (how do we keep them working when our customers are using them)

They develop four generic software business models based on a matrix. The vertical axis specifies how intensely the IT company collaborates with its customers and users; the horizontal how much they focus on standardised products rather than software custom-built for a particular situation.



This yields four types of business model:

- software tailoring – building tailor-made software for specific customers; example Logica developing the clearing system (CHAPS) for the British banking system
- applied formats – customized solutions based on common platform; example SAP offering solutions based on the its Enterprise Resource Planning (ERP) software
- resource provisioning – developing software components or middleware designed to integrate with other software; example Red Hat developing their own version of Linux and offering software and services around it
- standard offerings – own products sold widely and used without customization; example Apple selling integrated hardware and software products direct to private users.

They specify the four generic models in the table below[28]:

Business-model type	I 'Software tailoring'	II 'Applied formats'	III 'Resource provisioning'	IV 'Standard offerings'
Nature of offering	A typical offering is based on tailored, customer-specific solutions	Typical offerings embody customized solutions, based on a uniform core of several solutions or separate modules	The offering is based on a set of components, middleware or a product platform	Homogeneous offerings are based on a uniform core product, a modular product family or standardized online service
Customer-relationship construct	Customer relationships embody one-off production in close relationships	Customer relationship through value-adding resellers	Customer relationship through an internal hierarchy	Customer relationship through a wide distribution network including online distribution
Core competence	Ability to understand and meet customer-specific needs	Ability to understand and meet customers' needs within narrow segments	Ability to understand and meet technology-specific needs	Ability to serve common benefits of multiple customers

Though they are useful to stimulate thinking about different forms of software venture, it's not clear these generic business models go to the heart of what leads to competitive advantage in the software industry. For instance, Google, Microsoft and Apple all fall in the 'standard offerings' group - but their business models are radically different. A start-up positioning itself in the standardised offerings group would need a radically different business model, probably with a highly specialised market niche, in order to survive. Nor is it clear that one business model can describe all the activities in these companies; they don't with the four examples I used above.

What is clear is that engineering skills and the ability to develop software is necessary, but not sufficient for survival as a start-up. Chesbrough[3] argues that new technologies open up many possibilities for exploitation, and that it is not the technology itself, but the business model that exploits it which is the key ingredient in success. We'll look at open software business models later. New software companies need to develop viable business models relatively quickly; probably by thinking explicitly and acting proactively. They don't necessarily need to have a strategy or a document called a business model, but they need to make an explicit connection between the software applications and services they develop, and revenue generation and value in the business.

The software product

For many start-ups, the most important element is the software product. The really good product idea is often the inspiration and primary motivation for starting the company. The product needs to be differentiated from others in the market - you can't usually make a business by copying an existing product unless you intend to provide it much cheaper or for free (and derive income from

another source). In many cases the product will be innovative, maybe a spin-off from university collaboration or industry experience[15]. Often the survival of the firm is dependent, at least initially, on the success of the product. Heirman and Clarysse [29] argues that delivery of the first product is important for

- early revenues, improved cash flow and greater financial independence
- building a customer base
- engineering experience, and
- visibility in the market.

This also means that innovation speed - time to market - is important, especially where the company is heavily indebted to banks and venture capitalists. The longer a start-up works without income, the more difficult it is to achieve financial stability.

Case: Checkpoint

Checkpoint created an innovative process to build firewalls, security products that could go directly off the shelf to a customer and that enforce the boundaries between different networks and protect firms against unauthorised users. Checkpoint's programmers introduced a new language, Inspect, specifically for directing the rapid inspection of communication packets and a compiler to translate policy rules written in Inspect into assembly language. The program opens data packets, checks the content and quickly inspects each data packet. The innovation is that the program sends along the data in parcels after they are checked, rather than waiting to reassemble them before the entire transmission. This methodology increased dramatically the speed of data transmission, with the same level of security.[24]

Bringing the first product to market

How do you arrive at the first product? Companies typically start with an idea or concept, a software prototype or, more rarely, a finished product awaiting commercialisation. You'd imagine that starting with a lot of working code and a large amount of venture capital would be an advantage, but this turns out not necessarily to be the case. Software firms starting with a prototype often meet costly demands for redesign when they eventually meet with customers and users. Engineers preoccupied with software design often neglect market involvement and real experiences with customers. Those that are more successful involve customers from the beginning and this improves speed to market[29]. Neither is it automatic that having a large bag of money helps - this encourages gold-plating and lack of focus. Lean may be just as good. A good experienced partnership behind the software collaboration also improves speed to market. However Heirmann also points out that many software inventors have difficulty with shifting their mental models and adapting their brainchild to changing needs and conditions. Sometimes it may be important to believe in the original product concept and work at creating the market for it, but at other times flexibility is required to meet practical demands of users and customers, or react to competitor's products. Newly created small firms hardly ever have the resources and experience to operate traditional standardised development methods and are dependent on their ability to move flexibly forward through improvisation and bricolage[30]. Both the venture and its mode of working need to be agile. The first customer can be a really important link in developing a marketable product, providing much needed domain knowledge, input into requirements, cash injections (sometimes)

and the necessary motivation to finish a working application. De Haan and Cohen[30] point out that the traditional Systems Development Lifecycle (SDLC) is impossible to follow in entrepreneurial development. They suggest a 12 step process for off-the-shelf business software incorporating the first customer, which they call Lead-Driven Development. In this model the real development of the software is undertaken in cooperation with the first customer, whilst the development firm also identifies the kernel of the system which will be sold to a wider market. It is best-suited for the applied formats and resource provisioning styles of software company.

step	**description**
initiation – informal:	structuring the will and intent to begin the project, giving the go-ahead, providing limited resources for exploring the idea
high level concept development:	basic idea and scope, business value, developed in spare time sometimes with academic partners help, some code, no documentation or testing
demonstration prototype:	important milestone – partly functional prototype with good interface clearly addressing the problem that should be solved, many missing features, many bugs and engineering problem,
minor testing in non-profit environments, academic demo and use:	exploiting opportunities for showing and communicating application in non-commercial situations to achieve feedback on features, bugs and use
market introduction, benefit oriented demonstrations and mini pilots:	demonstrations at potential customers – customer specific: the development team tries to pick up as much information about the potential customer as possible and codes extra targeted features for their sales meetings. no documentation or test
offer product + implementer:	when a customer wishes to buy (first sale) they offer both the product and an implementer who will supervise the installation and train users
bug fixing:	the product is improved to a commercially useable 'non-frustrating' version targeted at the first customer's baseline requirements - many unresolved bugs remain
constrain features, further commercial releases, and support plan:	the development team identify a core set of features which will form the basis for further commercial releases (and attracting new customers) and develop a support plan for the software
develop complexity management tools, establish interface with customer's software, find additional benefit oriented projects, embed within organizational deliverables:	address complexity issues with first customer's work, synchronize software with customers' business goals, interface with other organizational systems, find additional customer problems to solve
second sale – licenses:	second sale to the same organisation and move to licensing arrangement indicates that the customer has moved beyond the learning stage and is ready to adopt the software
maintenance, begin user training, tutoring, software support:	here the customer begins to deal with the software without intensive support from the developers, who put more conventional training, maintenance and support processes in place
third sale, support	customer has moved beyond independently using the software and has realised the benefits of the product and committed to long term use

This kind of process will obviously not work for some other forms of software entrepreneurship where there is a software tailoring (pure consultancy) or standard offerings (generalized products for a broad market) business model.

Management

A business is also a leadership and management challenge. Running a business involves all kinds of housekeeping activities such as paying salaries and taxes, keeping accounts and customer information, and getting the building cleaned. You can easily end up doing some or all of these things because of the cost of employing others to do them. Past experience of management (finance, sales, technology, logistics, marketing, and selling, negotiating, leading, planning, decision making, problem solving, organizing, and communicating) is thought to be an advantage in a start-up[31]. As a software professional you possibly have no interest in many of these things, and no training in how to do them, which doesn't improve the chances of them being done well. Many of them can be outsourced. However, the team interaction of start-up founders in terms of leadership, interpersonal flexibility, team commitment and helpfulness are known to be important, and communication, cohesion, work norms, mutual support, coordination and conflict resolution are good predictors of technology venture success. Software development is characterized by a need to coordinate the work of individuals on day-to-day bases and cooperating teams achieve a higher level of integration of their individual interests and accomplish jointly the goals of the venture team[26].

The management of a small firm's software process is further question which can't be outsourced and is often neglected. Chenoweth[32] comments on the lack of conventional software engineering processes in start-up software companies. A freewheeling style may be inherent to software culture. He reflects that

> *'industry is motivated primarily by profit, and processes tend to be ignored by industrial organizations which are in a hurry. Start-ups in particular are creative, flexible and able to move quickly into opportunities because they are unencumbered by stifling layers of management. They have to move fast. Due to their lack of a customer base, knowledge about requirements does not come into a start-up in an orderly manner. As a result of this chaotic flow of requirements, any of the deliberate processes of software engineering can seem off-track in regard to keeping the new enterprise moving with that flow'.*

Chenoweth saw this lack of process in his industrial experience (technical diligence studies for a computer vendor and a telecom vendor) acquiring smaller software companies. Commonly, the winners in a first-to-market race had little documentation. Their requirements, designs and processes were largely in the heads of key people. Nevertheless he concludes that 'even in small start-up software businesses, experience, skills and methods are as important as new ideas in generating success.' A great idea cannot compensate for the absence of sound, appropriate engineering thinking about software development processes. He also characterises differences between the attitudes of young entrepreneurs in their first start-up and more experienced developers:

> *'start-up entrepreneurs we interviewed were characteristically impatient with even agile software engineering processes. This impatience was very clearly stated by the entrepreneurs as being due to their need to be first into the market, or to meet the needs of a key customer. The new businesses all added or changed processes during development as they became aware of issues. They especially changed how things were done in hopes of meeting their aggressive time-to-market expectations. The entrepreneurs' bias for action perhaps translated into the following logic: What has to be accomplished in order to reach the market? The most*

crucial ingredient, without which they have nothing, is the code itself. Everything else, therefore, is secondary in value. In contrast to the new entrepreneurs, discussions with the more established business people who had worked with these students showed a different attitude about process. A typical concern of the established businesses was that the processes and tools used on a new project at the incubator be like those used in other projects by that company. One established company brought-in their own on-site project manager to the incubator, specifically to help students use their processes and tools. Another company was there in order to use well-defined processes to reverse-engineer their existing, successful system which they had created without much process as a start-up, so as to begin building its successor'. [32]

These difficulties highlight the value of being able to learn rapidly from experience in order to combat the start-up's liabilities of newness and smallness [31].

Innovation speed and time-to-market

Time to market is a crucial factor for new ventures. The sale of the first product is a major milestone since it helps to gain early cash flow for greater financial independence, external visibility and legitimacy, to develop an early market share and increases the likelihood of survival[29]. Innovation speed is the time elapsed between initial development (including the conception and definition of the software or application), and ultimate commercialization, which is the introduction of a new product into the marketplace. Heirman[29] investigates several factors which are understood to increase innovation speed including: the starting resources of the company (which may be cash or specialist expertise), existing code or a prototype at company launch, the company's innovation processes, the experience and complementary skills (cross-functionality) of the development team and alliances with universities and R+D departments in industry. Surprisingly they found that

'starting with a beta version leads to significantly longer development efforts for software start-ups. Software firms starting with a beta version often face considerable reengineering issues that delay the launch of the first product. These software entrepreneurs developed their prototypes in the absence of—or at least a lack of—market information and lost precious time developing bells and whistles that customers do not want. Reengineering a beta prototype often takes longer than developing a software product from scratch, but right from the start. Software entrepreneurs should start their ventures early in the product development cycle, which enables them to freeze the concept in close interaction with the market.' [29]

My fellow entrepreneurs and I had been working in the graphical sector for a number of years. Two of us worked several years as software developers and the other one had many years of experience as sales representative. We worked for a company developing workflow automation systems for the graphical industry. When this firm was acquired, we felt unhappy with the new strategy and we talked about starting our own company. We brainstormed about potential business ideas for a couple of months. Our gut feeling was that printers needed software to automate their pre-press activities. We visited several printers and found out that they indeed were looking for tools to automate their pre-press activities and that none of the big players was focusing on this niche at the time. During these first meetings we let the customers explain what they exactly needed and how they wanted the product to look like. We said that we had a product on the shelf that could do

about half of that. With hindsight we took big risks, because we actually sold our first product before we developed it. We were not even sure that we could do it. We also promised that we would work on the other features they wanted if other companies had similar requests. We worked very hard to develop the software system that did the 50% job. We implemented this product with several customers. Later, we developed new versions including more features on customer's request.[29]

Software revenue generation models

If you want to survive as a software business you will need to generate some income. Many IT professionals imagine that this is quite simple – you write some software then you sell it, but the revenue generation models of many software companies are actually rather complex, often built up of many different income sources. You need to understand the value of your software for different groups of users, and structure your revenue generation strategies accordingly. Here are some of the major mechanisms:

- *license* revenue involves a contract to supply software for a predetermined number of users for a predetermined period of time. A license can be one-off and pay up front, upgrades can be free or paid for and in many situations there can be supplementary fees for implementation, tailoring and consultancy
- *subscription* revenue focuses on a time-boxed right to use software which expires if the subscription is not renewed: the payment can be expected to recur annually, quarterly or monthly
- *shareware* usually includes the right to trial the software (or a limited version of it) for a limited period, after which some form of license or subscription is required.
- *freeware* is available at no cost, or with an optional payment which is sometimes user determined. It should not be confused with free software because the copyright normally remains with the developers. Revenue is generated through a range of other mechanisms including advertising (which is sometimes targeted at particular user profiles), selling additional services (e.g. business analysis), tailoring, training and support. A short examination of free net-based applications and services (Google, Facebook, Wikipedia, MySpace) reveals many different related revenue generation mechanisms which become available where there are many 'eyes' (users)
- the *freemium* (free/premium) model usually offers a free starting pack, with a license or subscription for extra features and service levels
- *software as a service (SaaS)* involves the hosting of software and hardware facilities (with accompanying support services) for rent to customers – now often as cloud software delivered through thin clients or browsers. Payment is usually through some form of licensing or subscription agreement.
- *consultancy* fees are earned for delivering software made or tailored for specific situations, typically with accompanying implementation and support services
- *patent licensing* - revenue from the licensing of patents obtained
- *open revenue models* – concern the generation of revenue from open source software
- *platform access fees* are charged by companies that control a particular platform such as the IPhone apps market

This list covers some of the major ways of generating revenues from software, but there are many variations, and mature software firms' revenues are typically built up of many different revenue generation mechanisms. Start-ups need to be clear where they will begin.

Entry and growth

An existing industry, in Porter's characterisation, provides barriers to entry. This means that your competitors do not intend to let you take some of their customers and will work hard to avoid it. Start-ups have the disadvantages of smallness and newness: few engineers with many tasks, little experience, no existing products, reputation, brand or customers and so on. Established firms already have these things, plus easier access to capital to invest in new projects, the robustness and cash flow that allows them to survive the occasional fiasco, mature engineering practice and wider skills profiles (they don't have to find out about good accounting practice). This means that entry into an industry can be a problem. Ojala and Tyrväinen[33] argue that the drivers of software start-up entry are

- innovation (the software product, algorithm or technique at the leading edge)
- capabilities of entrepreneurs (including both their specialised engineering competences, and their entrepreneurship abilities)
- patents (protected intellectual capital)
- specialization in a specific market niche

Giarattana[24] argues that the drivers of start-up growth are primarily product differentiation and international expansion. Software is a universal language with many shared elements in hardware platforms, and expansion to other markets can be as simple as translating the interface. Heavily customized products can be harder to internationalise, because they relate to particular cultures and work practices.

Ojala and Tyrväinen consider entry into new markets abroad and argue that the choice of entry mode for a software start-up is connected to their business model and primarily to the product strategy.

Networks, partnerships and alliances

Another important component of growth is the development of networking and partnerships (technological alliances). Studies of start-ups and of spinoffs show extremely complex patterns of relationships and contacts between software firms. Giarratana shows a network of 256 partnerships and alliances between firms during the birth of the encryption software industry in Silicon Valley[24]. Small firms typically have specialised competences and have difficulty acquiring the skills necessary to stay abreast of technology developments and to bring finished products to market. They may choose to outsource software development which is not in their core competence area (such as interface design, mobile programming or usability testing). Technological alliances bring complementary expertise and existing code bases, and can help determine the innovativeness of products. For start-ups, an alliance with an established IT partner can greatly simplify the process of entering a market.

Heirman suggests that collaborations with universities help start-ups to stay at the technological leading edge, whilst collaboration with industry partners open up resources and new markets. Universities are often fertile grounds for entrepreneurship, where new ideas are developed through classroom projects[34] and research, without the additional risk of commercial failure. Many universities also have well-developed systems for moving ideas into the commercial arena through incubators and university-sponsored companies.

Case study: Certicom

Certicom's Elliptic Curve Cryptography is a technology especially useful in "small-footprint environments" such as smart cards or wireless communications devices, where space is the scarcest resource. If the standard string of computer bits necessary to encode or decode an encrypted message needs about 1,024 bits, Certicom's system accomplishes in 160. The difference is rooted in mathematics. In fact while the standard cryptographic systems are based on integer calculus, the elliptic curve cryptosystem uses equations that can be calculated more easily and faster.[29]

Shane[35] points out that venture capitalists are heavily influenced by social network connections (particularly direct social ties) and the personal reputations of the entrepreneurs that come to them with proposals. It's said that an entrepreneur should never eat dinner alone:

> *'I had met [ENTREPRENEUR] in the 1970s when I got involved in financing some technology start-ups in Israel. I ended up funding [ENTREPRENEUR's] company and I joined the board. I worked very closely with [ENTREPRENEUR] and his staff and we became close personal friends ... I sold my stock when he left because my relationship was with him personally. We stayed in touch and remained good, close, personal friends. When he came to the United States, we started to discuss his ideas for [COMPANY]; [ENTREPRENEUR] asked me to be a founding investor of the company. So I put in some money and joined the board.'*[35]

Hung[36] reinforces the value of social capital in helping the entrepreneur, whether it results from family connections or earlier business relationships. This way of thinking is also central to Sarasvathy's thinking and the Consider, Do, Adjust paradigm with its focus on building new relationships and commitments.

Steve Chang founded Trend Micro using funding from family group members. He had little prior business experience but a strong network of family and friends with similar backgrounds in the local community, who supported him.

Morris Chang (the founder of TSMC Taiwan Semiconductor Manufacturing Company) possessed international human relations from his extensive international business experience. This dominated TSMC's development. The original capital of TSMC was sourced from governmental financial support and other domestic and large-scale overseas companies' investments. Moms Chang's social relations derived from an extensive array of contacts with past colleagues, experts and advisers; many derived from past commercial relations with customers, suppliers and banks.[36]

Internationalisation

New high technology venture benefit from internationalisation (expanding to foreign markets) through learning about new technologies, combatting fiercely competitive home markets,

commercialising their products and maximising profitability[37]. Internationalisation gives new ventures access to valuable resources, solidifies their competitive positions and improves their performance, though success is limited by various constraints, including personnel and production capability, insufficient knowledge about markets and customers, weak financial resources to support growth, and poor skills in internationalization and marketing[18]. Software innovations diffuse rapidly, partly because of piracy in countries with weak intellectual property laws, and the benefits to the original innovator developers are lost.[37] Another challenge facing high-technology firms is the decrease in the length of the technological life cycle, and the necessity to get their products and services into global distribution within a limited window of opportunity. Start-ups usually have the disadvantages of smallness and newness to overcome, but software is innately international, both because programming languages are based on English-like constructions and are commonly understood, and because the mechanism of distribution (chiefly the internet) is extremely cheap. If you are starting a software company you should have a very good reason not to think internationally. The potential user market for a Danish language mobile application is about five million; the market for the same application with an English language interface is 95m in the US alone. The cost of translating the interface is negligible. China now has 900 million mobile users and in 2020 the number of English-speaking Chinese mobile users is expected to roughly equal all other English-speaking mobile users.

Partnering is important for making international progress, and partners can fulfil various roles, including

- system integrator - provides consultation for the end-users (defines their needs) and designs custom solutions
- solution provider - work is based on the end-user's definition of needs
- (value added) reseller - provides products with configuration and integration, turn-key projects
- volume distributor - distributor in the chain, mostly usable for packaged goods/software products
- retailer - business front-end sales partner
- sales agent/representative - third-party software vendor, revenues based on fees from the actual sales
- independent software vendor software provider without contractual relationship with you
- influencer, consultant etc. companies that comment, evaluate, and give guidance and advice to end-users
- original/own equipment manufacturer - normally provides privately labelled product.[18]

SOFTPRO (a fictitious name) started as a product development unit of a parent corporation (a Finnish telecommunications company) before starting to sell its products to external customers. The product of the company is a systemic software product for business customers with a standardized core suitable for international customers. However, the products require adaptation and integration - normally carried out by local partners. The company operates worldwide using partners in the implementation, installation and customization of its products, which is typical of solution-provider businesses. It has traditionally used external partners in its internationalization process as a channel to new markets. Half a dozen value-added resellers form a core network for the company's international operations, and new partners are sought in the established partner-choice process.

Operations started in 1990 when the first version of the software was released. The first international sales occurred in 1994 and the software is now licensed to more than 250 customers around the world, Germany and France being the key markets. The company also has a value-added reseller in Australia, and some operations in the US. Its business-communication solution covers most industry sectors and most of the end-customers are in retailing, transportation and logistics. The firm relies strongly on its partners and has concentrated its own efforts on product development, as well as on partner training and support. The selling, the consultancy work, the integration of the software into customers' other applications and the customer training are normally done by the partners. Several key players in the network create value for the end-user, and the success of the product in general depends on the co-operation of all the members in the network. Partnering is seen as a fast and efficient way of doing business; sales go through partners with local knowledge and contacts.

SOFTPRO has well-defined *partner-selection* processes with written criteria including (1) business criteria, (2) marketing criteria, (3) partner-potential criteria and (4) technical criteria. The *partnering process* includes discussions in which it is defined how well a channel partner candidate matches the profile. The screening process is demanding, and finding a good partner is difficult. Their willingness to develop their business is also discussed. In *partner evaluation* the fit of the potential partner is evaluated on four dimensions: (1) product and service, (2) customer, (3) marketing and sales and (4) business potential. The most important aspect is the ability to offer a complete chain of customer-service[18].

Finance: venture capital or bootstrapping

In the glamorous world of Silicon Valley, the primary source of finance for software start-ups is venture capital. In the Analyse, Design, Enact paradigm the turning point in starting a business is raising the capital to finance it. This involves convincing an experienced banker to bet on the success of the venture. Kaplan describes a dizzying series of meetings and telephone calls with potential backers, demonstrations with hardly functioning handwriting recognition software, and brilliantly improvised presentations, before Go Corporation raised their initial funding of $6M (about DK 100M in today's money). Six years later they had spent $75M, without selling a single copy of their software, or generating any return for their investors. However the conventional tool of persuasion is the business plan, and the need to raise capital is the single compelling reason for developing an effective one. Most entrepreneurs will want to raise loan capital at some point, because this investment greatly increases the range of actions that can be taken to develop the business (new software projects, new engineers, professional business guidance, marketing etc.)[35]. It often significantly reduces the entrepreneur's stake in the company as venture capitalists take large stock holdings to offset their risk. However, there are other ways of getting a software firm off the ground. Some entrepreneurs make a conscious choice to avoid or delay raising venture capital in order to preserve the value of their equity stake [38]. Most are simply unable to raise finance from external sources due to:

- lack of innovative or original ideas
- limited industry and management experience
- inability to convince investors that the business concept can make a profit

In these cases entrepreneurs use bootstrapping: 'imaginative and parsimonious strategies for marshalling and gaining control of resources' [38]. These are more likely to be associated with a Consider, Do, Adjust paradigm approach to building the start-up. The resource-base of the firm can be understood as:

- human capital - the software professionals (and others) involved
- intellectual capital - the specialised competences of these professionals
- social capital - the commitments that start-up members make to each other through the resources they invest
- physical capital - primarily hardware and development environment software
- financial capital - the funds needed to initiate and grow the business

Bootstrapping strategies can help with most of these. A common strategy is using consultancy work to finance own product development and other aspects of developing a start-up. Many other strategies focus on securing resources at little or no cost:

- investing your own time (for instance developer hours)
- using open source code instead of writing your own
- working with university incubators and greenhouses
- using social contacts, for instance getting someone to help who owes you a favour
- lean management and development techniques
- informal collaborations (swap software components, shared development)
- borrow office space
- use of free software tools (subversion, SCRUMdo, Google docs) to save resources
- research collaborations for knowledge generation
- combining forming a company and developing a product with study.

Bootstrapping techniques for product development include: special deals for access to hardware, development in the evenings and weekends alongside a day job, research grants, customer-funded research and development, commercializing university-based research, commercializing public domain software, porting fees to transfer software from one platform to another, free or subsidized access to hardware, commercializing an existing shareware product, turning a consulting project into a commercial product and using public domain development tools.

Bootstrapping techniques for business development include:

- delaying payments to suppliers
- barter arrangements
- personal credit cards & home equity/mortgage loans
- discounted advance payments from customers
- below market or very low rent space
- deals with professional service providers at below competitive rates
- leasing vs. purchasing assets
- purchasing used vs. new equipment
- working out of home
- gifts or interest-free loans from relatives
- unpaid family member working as an assistant
- severance and parachute payments

- personal savings
- reduced compensation
- forgone or delayed compensation
- special terms with customers, including discounted advances, pre-payments and larger than normal deposits
- outsourcing key parts of the business
- shareware revenue stream [38]

Harrison suggests that growth-oriented and growth-achieving software firms will be more likely to use extended value chain collaborative relationships as bootstrapping techniques rather than cost-reduction relationships.

Bootstrapping and venture capital do not necessarily need to be alternatives – bootstrapping is sometimes necessary to support the early development of a venture before it can be attractive for investors. The angels and venture capitalist investment can come later.

Sources

DE HAAN, U. & COHEN, S. (2007) The role of improvisation in Off-the-Shelf software development of entrepreneurial vendors. *2007 International Conference on Systems Engineering and Modeling, Proceedings*, 85-92.

GIARRATANA, M. S. (2004) The birth of a new industry: entry by start-ups and the drivers of firm growth - The case of encryption software. *Research Policy,* 33, 787-806.

HARRISON, R., MASON, C. & GIRLING, P. (2004) Financial bootstrapping and venture development in the software industry. *Entrepreneurship & Regional Development,* 16, 307-333.

HEIRMAN, A. & CLARYSSE, B. (2007) Which tangible and intangible assets matter for innovation speed in start-ups? *Journal of Product Innovation Management,* 24, 303-315.

HUNG, S. & HSIAO, Y. (2004) Mobilizing social capital to pursue entrepreneurship.

IGEL, B. & ISLAM, N. (2001) Strategies for service and market development of entrepreneurial software designing firms. *Technovation,* 21, 157-166.

MANN, R. & SAGER, T. (2007) Patents, venture capital, and software start-ups. *Research Policy,* 36, 193-208.

MUELLER, T. & GEMUNDEN, H. (2009) Founder team interaction, customer and competitor orientation in software ventures. *Management Research News,* 32, 539-554.

OJALA, A. & TYRVÄINEN, P. (2006) Business models and market entry mode choice of small software firms. *Journal of International Entrepreneurship,* 4, 69-81.

PAULI, J. W., LAWRENCE, T. E. & BROWN, B. F. (2008) Development of a new software product from a classroom project. *Proceedings of the Fifth International Conference on Information Technology: New Generations*, 97-100.

RAJALA, R., NISSILÄ, J. & WESTERLUND, M. (2007) Revenue models in the open source software business. *Handbook of Research on Open Source Software: Technological, Economic, and***,** 541.

RAJALA, R., ROSSI, M. & TUUNAINEN, V. (2003) A framework for analyzing software business models. Citeseer.

RAJALA, R. & WESTERLUND, M. (2007) Business models a new perspective on firms' assets and capabilities: observations from the Finnish software industry. *The International Journal of Entrepreneurship and Innovation,* 8**,** 115-126.

ROSE, J. (2010) *Software Innovation: eight work-style heuristics for creative software developers,* Aalborg, Software Innovation, Dept. of Computer Science, Aalborg University.

VARIS, J., KUIVALAINEN, O. & SAARENKETO, S. (2005) Partner selection for international marketing and distribution in corporate new ventures. *Journal of International Entrepreneurship,* 3**,** 19-36.

YINGYU, D. & YE, W. (2008) The Mechanisms of Learning and the Survival of New Ventures.

ZAHRA, S., MATHERNE, B. & CARLETON, J. (2003) Technological resource leveraging and the internationalisation of new ventures. *Journal of International Entrepreneurship,* 1**,** 163-186.

e-Entrepreneurship

Whereas the business model of a software start-up is primarily concerned with writing software, many other entrepreneurial opportunities exist for software professionals in the net economy. Here we will consider other forms of business which are enabled by software. Classic examples are Facebook, eBay, Amazon. Here the business models revolve around social networking, auctions and book-selling, and the products are information products. However they are net-companies enabled by its software - without software they doesn't exist and their engineering teams deal with complex software issues, particularly scaling issues. They are therefore software dependent. These platforms have mobile extensions, and there are also emerging mobile businesses; these two markets are heavily interrelated and many similar entrepreneurial opportunities arise. Most businesses (and public administration) are supported to some extent by software, so dependence is a question of degree. Software dependent start-ups are also suitable ventures for software professionals. Here you have, or can acquire the software competencies; what's missing is the business idea. It's not really necessary to be entirely located in the net-economy to be software dependent, many information-rich kinds of businesses such as banks and insurance companies are also software dependent.

Many of the big net-economy players began as a start-up, with an entrepreneur or entrepreneurial team composed of engineers with computer science degrees and backgrounds as engineers in software companies (many of them have histories as software developers and experience as entrepreneurs before the major companies that they are associated with):

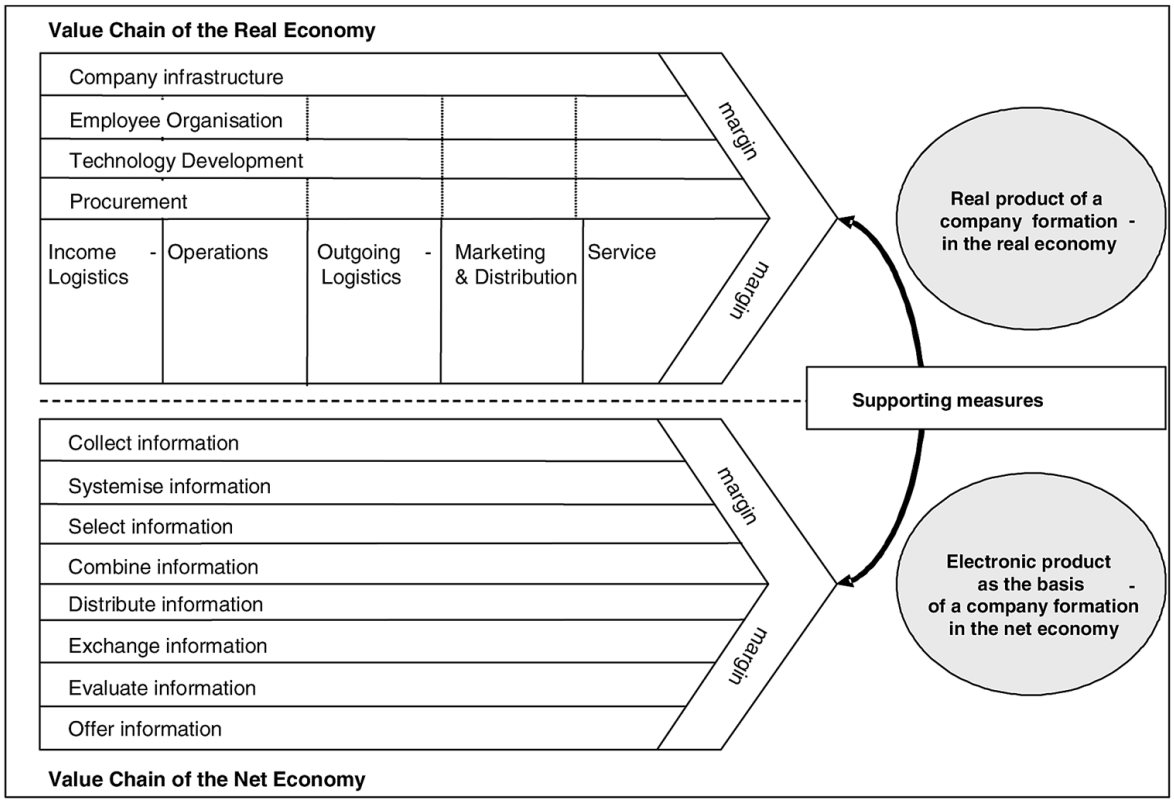
- Pierre Omidyar - Ebay
- Jeff Bezos - Amazon
- Mark Zuckerberg -Facebook
- Chad Hurley, Steve Chen, Jawed Karim - YouTube
- Evan Williams -twitter

The developing mobile services market shares many of these characteristics. Nor is it clear that there are strict boundaries between being a software start-up and being part of the net-economy. A company developing iPhone apps earns a living by writing software, but each of the apps must have a separate value proposition for the user that means that they will pay for it.

The net economy

The net-economy has some special characteristics which distinguish it from the conventional economy, for which many of the reference theory models presented earlier were developed. Kollman[39] argues that it has three building blocks: information, communication, transaction. There are several forms of electronic commerce including eProcurement, eShop, eMarketplace, eCommunity, and eCompany. Kollman contrasts the value chain of the conventional economy (best suited for conceptualising the transformation of physical raw materials into valuable products) with the information transformations of the net economy value chains. The less involvement with physical products and their manipulation and logistics, the more software dependent is the company. Companies which have only digitalised information products (including, for example a net bank) are fully software dependent. Basic value chain activities include collecting, systemising, selecting,

combining, distributing, exchanging, evaluating and offering information. However Kollman's use of the word information is too restricting in this context; anything that can be digitalised, represented as bits, stored on a computer and transferred over a network is part of the net economy. This includes picture, music and video files and streaming services (and many other things that we don't normally call information).



Value creation takes place through:

- Overview: the structuring of large amounts of digital information.
- Selection: the ability to locate desirable information (e.g. the product you wish to buy)
- Transacting : enacting an agreement between two parties, for instance a purchase between buyer and seller
- Cooperation: the online linking and matching of products and services (buy a flight and book a hotel at your destination).
- Exchange: consumer information exchange and communication (e.g. customer review, rating and recommendation)

Ziinga.com is a commercial internet-based business using the penny-auction model. Attractive prizes are offered, many of them desirable high-tech artefacts such as iPads and iPhones. Users register and are them allowed to bid for the products using a specialised auction model. Bids are not expensive (hence the 'penny' title) and are placed in fixed time intervals – usually between 20 seconds and two minutes. A visible clock counts the time period and the bidder with the last bid if the clock hits zero without a competing bid coming in wins. The price of the prize starts very low and increases in rather small increments at each new time period – the winner pays only the indicated auction price and the cost of their bids. The business is primarily digital, the only physical processes concern transferring prizes to the winners.

The value proposition for users is the opportunity to win a desirable prize at a fraction of its normal retail value. An element of skill is involved: many will bid when the price is low and fewer when it gets too high so a bidder must know (guess) when it is optimal to bid, and bidding can be automated with a bot to help this process. If the user gets tired of bidding they can buy the product at market price, offsetting the cost of their bids. A variety of devices are used to make winning seem easier than it really is; for instance is not evident that the bidder is competing internationally with many other bidders in many countries. A penny auction provider can easily manipulate the auction with its own bots, but this is illegal in many countries. Revenue is generated primarily by ensuring that the cost of the bids purchased by many users is greater than the costs of buying and transporting the prizes and all the associated business costs: this can be actuarially determined.

eCommerce business models

Michael Rappa http://digitalenterprise.org/models/models.html has done quite a lot of work on generic business models for eCommerce (see insert). These can help with structuring thinking about e-business ventures. His categorization scheme is included for completeness, but you should notice that it includes things which overlap with discussions elsewhere in the notes (for instance software start-ups, revenue generation models and open source business models). It illustrates how many different types of opportunities are available in this field.

business model	Type	description
brokerage	marketplace exchange	full range of services covering the transaction process, from market assessment to negotiation and fulfilment - exchanges operate independently or are backed by an industry consortium

	buy/sell fulfilment	takes customer orders to buy or sell a product or service, including arranging terms (e.g. price and delivery)
	demand collection system	the patented 'name-your-price' model pioneered by priceline.com - prospective buyer makes a (binding) bid, broker arranges completion
	auction broker	conducts auctions for sellers (individuals or merchants) - broker charges the seller a listing fee and commission based on the value of the transaction – many forms of auctions with different offering and bidding rules
	transaction broker	provides a third-party payment mechanism for buyers and sellers to settle a transaction
	distributor	catalogue operation that connects a large number of product manufacturers with volume and retail buyers - broker facilitates business transactions between franchised distributors and their trading partners
	search agent	software agent used to search for the price and availability of goods or a service specified by the buyer or to locate hard-to-find information
	virtual mall	a hosting service for on-line merchants that charges setup, monthly listing, and/or transaction fees - may also provide automated transaction and relationship marketing services
advertising	portal	usually a search engine that may include varied content or services - high volume of user traffic makes advertising profitable and permits further diversification of site services – can be personalised
	classifieds	list of items for sale or wanted for purchase - listing fees, sometimes membership fee
	registered user	content-based sites that are free to access but require users to register - registration allows inter-session tracking of user surfing habits and thereby generates data for targeted advertising campaigns
	query-based paid placement	sells favourable link positioning (i.e., sponsored links) or advertising keyed to particular search terms in a user query
	contextual advertising	freeware developers who bundle ads with their product e.g. browser extension that automates authentication and form fill-ins may also deliver advertising links or pop-ups as the user surfs the web
	content-targeted advertising	identifies the content of a web page and then automatically delivers relevant ads when a user visits that page - pioneered by Google
	ultramercials	interactive online ads that require user interaction to reach the intended content
information intermediary	advertising networks	service feeding banner ads to a network of member sites, thereby enabling advertisers to deploy large marketing campaigns – collect data about web users that can be used to analyse marketing effectiveness
	audience measurement service	on-line audience market research
	incentive	customer loyalty programs providing incentives to customers

	marketing	such as redeemable points or coupons for making purchases from associated retailers - data collected about users are sold for targeted advertising
merchant	virtual merchant	a retail merchant that operates solely over the web ('e-tailer')
	catalogue merchant	mail-order business with a web-based catalogue which combines mail, telephone, and on-line ordering
	click and mortar	traditional brick-and-mortar retail establishment with a web storefront
	bit vendor	merchant who deals strictly in digital products and services and, in its purest form, conducts both sales and distribution over the web
manufacturer direct	purchase model	manufacturer that sells its products or services directly to the consumer
	lease model	manufacturer that finances the sale or rental of its products directly to the consumer
	licensing model	manufacturer, such as a software maker, that licenses its product directly to the consumer
	brand-integrated content	created by the manufacturer for the sole purpose of product placement
affiliate	banner exchange	trades banner placement among a network of affiliated sites
	pay-per-click	site that pays affiliates for a user click-through
	revenue sharing	offers a percent-of-sale commission based on a user click-through in which the user subsequently purchases a product
community	open source	software developed voluntarily by a global community of programmers who share code openly - instead of licensing code for a fee, open source relies on revenue generated from related services like systems integration, product support, tutorials, and user documentation
	public broadcasting	user contributor model used by not-for-profit radio and television broadcasting extended to the web. the model is based on the creation of a community of users who support the site through voluntary donations
	knowledge networks	discussion sites that provide a source of information based on the sharing of expertise among professionals
subscription	content service	provides text, audio, or video content to users who subscribe for a fee to gain access to the service
	person-to-person networking service	conduit for the distribution of user-submitted information, for example, individuals searching for former schoolmates
	trust service	membership association that abides by an explicit code of conduct and to which members pay a subscription fee
	internet service provider	provides network connectivity and related services
utility	metered usage	measures and bills users based on actual usage of a service
	metered subscription	allows subscribers to purchase access to content in metered amounts (e.g. numbers of pages viewed)

Many ecommerce business models have grown out of pre-internet ways of doing business; for instance Amazon is a net-economy version of the traditional book-selling industry. The bricks and mortar high-street retailers are removed from the traditional supply chain, making the eCommerce value chain cheaper to operate than the traditional. Removing a link from the value chain is known as disintermediation.

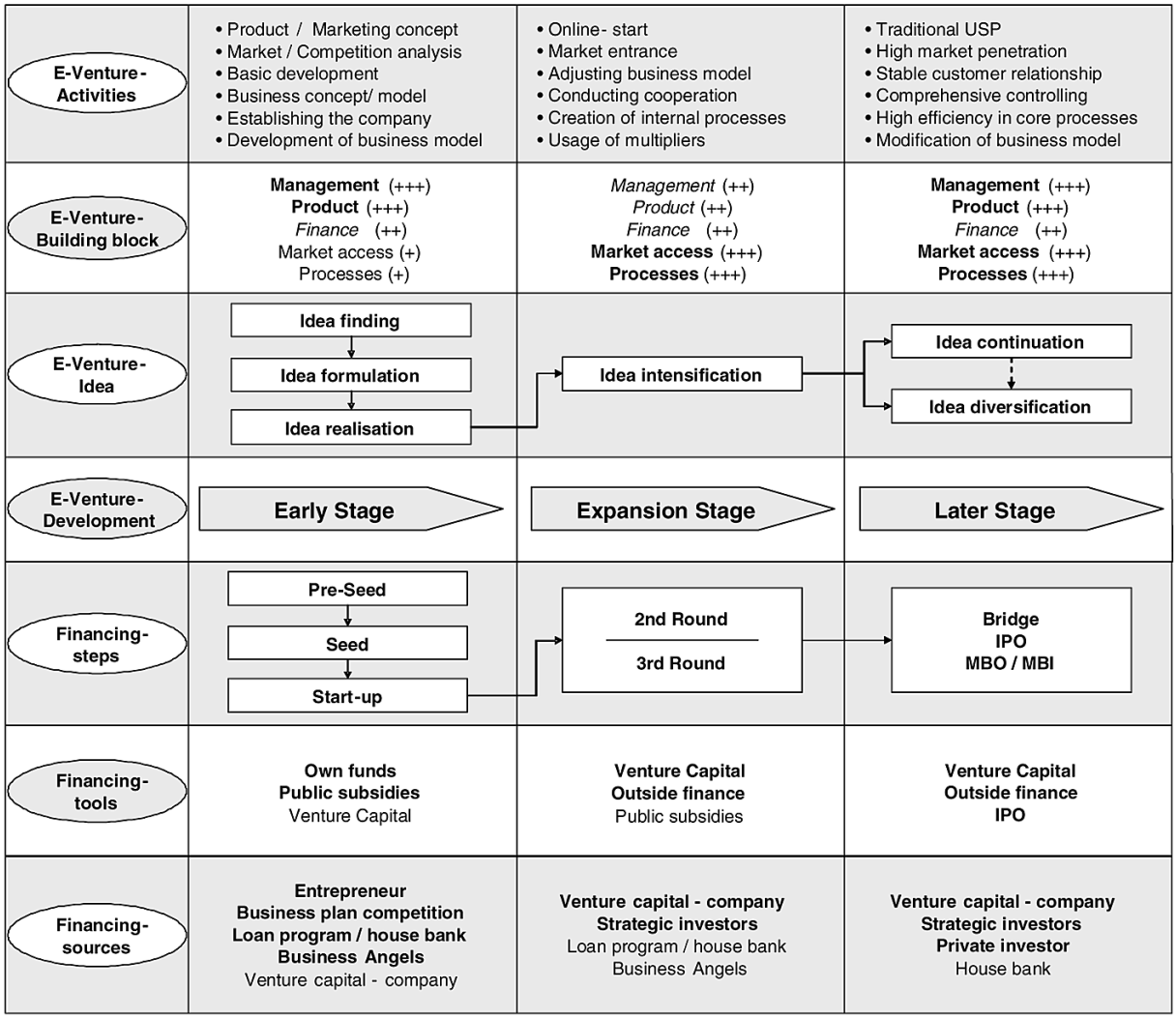
The transformation of the publishing industry: Lulu.com

The traditional publishing industry relies on highly professional authors who submit their work to publishing houses who carefully screen and select. The editing and printing and promotion processes are expensive and time-consuming and the publisher is dependent upon selling relatively large numbers of books to retailers and bookstores to make a profit.

Lulu.com turned the traditional bestseller-centric publishing model on its head by making it possible for anyone to publish. Niche and amateur authors can bring their work to market, using their own word-processors. It eliminates traditional entry barriers by providing authors with the tools to craft, print, and distribute their work through an online marketplace. Physical copies are printed on-demand as they are ordered, e-books are simply downloaded. Lulu generates micro revenues from each sale and provides value-added services, such as tailored help with editing, distribution and marketing. The failure of a particular title to sell is irrelevant to Lulu, because such a failure incurs no costs[4].

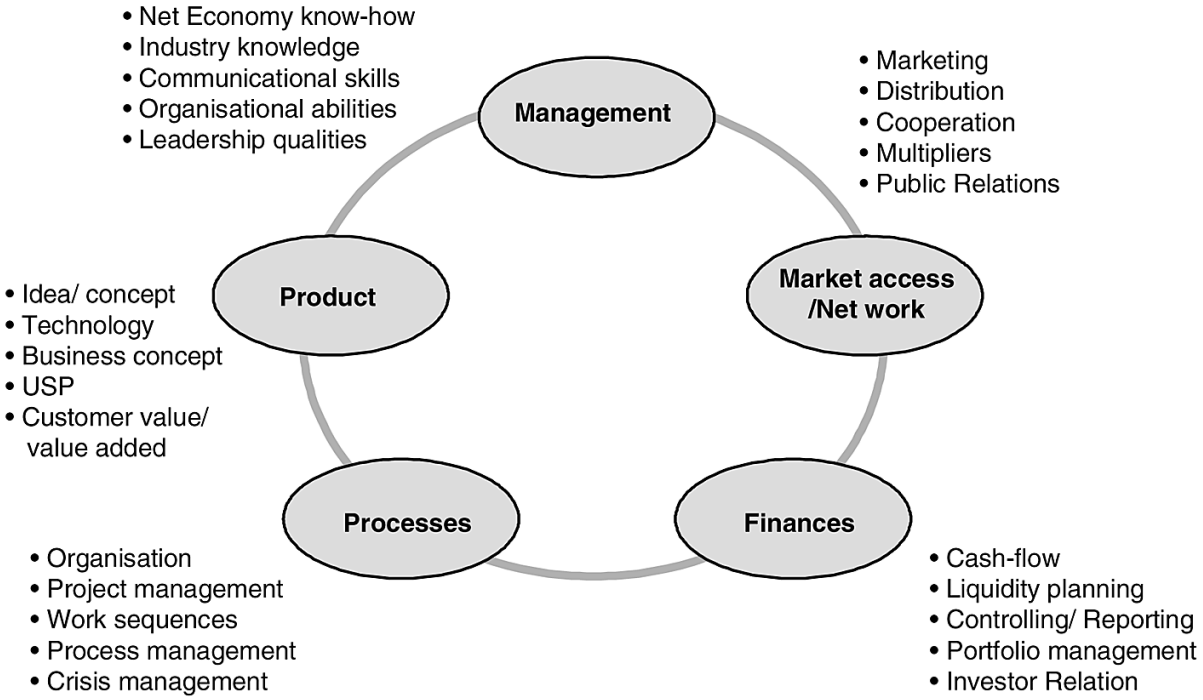
Starting an e-venture

Kollman provides the following framework identifying three stages of e-venture development[39]. Three stages of growth are conceptualised: early, expansion, late. The activities in each stage are characterized, some building blocks specified and prioritised for the stages, (management, product development, finance generation, market access, establishing business processes) together with a focus on the economic aspects of entrepreneurship (financing steps, tools, sources).



eCommerce success factors

Kollman identifies success factors for e-ventures as[39]:



Koller's model for e-venture development is both located in the analysis-heavy ADE tradition, and primarily business-focused. The skills of the software engineer are not really mentioned or prioritised and it seems as though he assumes that producing and maintaining the software platform which will underpin the net business is a trivial task hardly worthy of mention.

Mobile ventures

Mobile (-m) entrepreneurship share many of the characteristics of e-entrepreneurship; for example the principles of working with digital products and services which can be delivered direct to the device they will run on.

> *'the nature of the mobile applications market provides ample opportunities for entrepreneurial activity. To exploit these opportunities mobile application developers must generate successful market entry strategies, and among them a niche market strategy is considered the first step towards financial gain. Almost all entrants pointed to the unique space, market position, and requirement they had carved out for engaging in business…… in order to be able to reap the benefits of economies of scale application deployment has to cater to a broad variety of devices, operating systems, and networks'*[40]

There are also some special considerations. The structural conditions of the market dictate that new entrants must

- cope with heterogeneous platforms (the considerable challenges of delivering products to many different hardware platforms with somewhat different capabilities, served by different operating systems where at least one of the OS's (Android) has many variants, and
- comply with proprietor market standards (making sure that products satisfy the conditions set by the market operator (e.g. Apple's app market).

Mobile business models

It's also clear that, in the same way that the business models of eCommerce emerge from those of bricks and mortar commerce, many mobile applications and service reflect the trends of the eCommerce world. Thus it's not easy to build a mobile social network service from scratch, because it must compete with a Facebook smartphone application which couples the mobile network together with the many millions of existing users on the conventional internet.

Nokia Maps

Nokia Maps is a service for selected Nokia handsets. It offers free mapping and routing services for mobile device users to find the way to their destinations. Besides, it provides information about points of interest like tourist interest, restaurant and hotels. In addition to these free services, premium services including city guide and voice navigation are available for handsets with GPS capabilities.

Maps doesn't currently generate any revenue for Nokia (if the navigation service is not taken into account). According to Nokia, however, there are three potential sources of future revenue for its location based services besides navigation. It could serve as an advertising channel a la Google Maps, or charge referral fees from businesses which connect to Nokia users through the handset vendor. Another possibility is to sell premium contents (e.g. GPS waypoint coordinates, point of interest location information) to Maps users.

Mobile software products and services are not simply cut down versions of eCommerce applications and services and it's not yet really clear whether there are specialised mobile business models. In any case products must

- recognise the limitations of mobile applications (small screen, memory limits, relatively high costs of data exchange……), and
- exploit mobility features (e.g. context awareness) and smartphone hardware technologies (GIS, touch screen, motion sensors magnetometer.....)

Recent research into common mobile business models[41] is summarised in the following table:

mobile business model	service examples
Timeliness	stock information, news, sports information
remote access	intranet access, integrated messaging, banking/trading, reservation, sales support
remote control	information appliances, automobile application, navigation tracking, emergency service
location-based	traffic information, weather information, vehicle location, entertainment
personal communication	messaging, personal directory, chat, email community
mobile commerce	ticketing, usage fee, banking/trading, auction
business application	conference support, intranet access, file transfer, database access
Entertainment	music, game, graphic, video, TV

Mobile service design

According to Faber[42], there are four components of designing a mobile service:

- service design describes a firm's service offering to specific customers/end users in a particular market segment with a focus on intended and perceived value
- organisation design describes the configuration of actors (value network) possessing certain resources and capabilities, which together perform value activities to create value for the customer
- finance design describes how a company intends to generate revenues from a particular service offering including: financial arrangements, revenues, costs, risks and investments
- technology design describes the technical architecture and functionality that is needed to realise a certain service offering.

Since the business elements have been in focus earlier in this section we'll have a look in more detail at technical requirements for mobile services. The important technical design considerations described by Faber are summarised in the following table.

component	description	design considerations
technical architecture	overall architecture of the components listed below	centralised or distributed
		open or closed
		interoperable or non-interoperable
backbone infrastructure	long- and medium range backbone network infrastructure	high or very high bandwidth
		future-proof or non-future-proof
access networks	first and second mile network infrastructure	fixed or wireless
		high or low bandwidth
		universally available or deployed in hotspots
		scalable or non-scalable
service platforms	middleware platforms enabling different functions (e.g. billing, customer data management, location information)	centralised or distributed
		personalised or non-personalised
		secure or non-secure
		legacy or new
		open or closed
devices	end-user devices providing access to services	multi-purpose or single-purpose
		'network intelligent' or 'dumb interface'
		storage facilities or no storage facilities
		embedded software or open terminal
applications	the user applications running on the technological system	communication or content
		always on or time-critical
		personalised or non-personalised
		secure or non-secure
data	data streams transferred over networks	bursty or real-time
		high volume or low volume
technical functionality	functionality offered by the technological system	always on or time-critical
		personalised or non-personalised
		secure or non-secure

Mobile entrepreneurs often have to adapt to the demands of micro-development (many small apps adapted to different hardware and software platforms) where the average return is relatively low. Real hits (Angry Birds) with six-figure sales are rare, and a high proportion of downloaded apps are only ever used once. Thus means that mobile entrepreneurs may have to build their businesses on portfolios of micro product revenues. On the positive side, the entry barriers for mobile entrepreneurs are extremely low – you cam programme your first apps in your living room in your spare time after work. It's also becoming clear that the mobile phone is the communication devise of choice in developing countries where there is no conventional internet infrastructure. Infrastructure development can simply hop over the conventional PC internet browser and go straight to the mobile device. As more smartphones become available the potential markets are enormous.

Software focus, with Consider, Do, Adjust

E-and m-entrepreneurship are often discussed in the literature with a business focus, and in the light of the huge successes (Facebook, eBay, Amazon) where an Analyse, Design, Enact perspective is assumed. However Omidyar makes it clear in his comments on the founding of eBay that this assumption is not necessarily correct:

> *'If I had had a blank check from a big VC, and a big staff running around - things might have gone much worse…… I had to operate on a tight budget……necessity focused me on simplicity: eBay was open to organic growth ……whatever future you're building… don't try to program everything……. build a platform……. prepare for the unexpected'*

His account is both effectual (CDA) and focused on building software. The attraction for software and IT professionals lies in the relatively easy start-up options. If you have a true software-dependent idea for a digital business or product and you can write software and manage a software platform, then start-up costs can be quite low. For some mobile opportunities the entry barriers are extremely low – just write your app. Engineers have wide experience with software products and a good instinctive understanding of technology trajectories and what certain (young, tech-oriented) market segments could find interesting. However the problem here may be generating or recognising the business idea; there's not much in an engineer's education that encourages these skills. A further challenge lies in evolving the business model aspects of the idea so that the product or service can establish itself as financially viable.

Sources

FABER, E., BALLON, P., BOUWMAN, H., HAAKER, T., RIETKERK, O. & STEEN, M. (2003) Designing business models for mobile ICT services. *16th Electronic Commerce Conference.* Bled, Slovenia.

KOLLMANN, T. (2006) What is e-entrepreneurship? - fundamentals of company founding in the net economy. International Journal of Technology Management, 33, 322-340.

LEEM, C. S., SUH, H. S. & KIM, D. S. (2004) A classification of mobile business models and its applications. *Industrial Management & Data Systems,* 104**,** 78-87.

TARNACHA, A. & MAITLAND, C. F. (2006) Entrepreneurship in mobile application development. *2006 ICEC: Eighth International Conference on Electronic Commerce, Proceedings*, 589-593

Open entrepreneurship

Open entrepreneurship has become an interesting topic after the success of many companies (Red hat, Canonical, Mandriva) and the widespread incorporation of open source development into mainstream companies business models (Sun, Microsoft, Apple). The open source movement is itself interesting because making source code freely available contradicts a standard premise of conventional software business models - that you should protect your code to prevent other companies copying it and thereby gaining access to your markets.

> *"What has startled programmers and academics alike is the surprising success of OSS projects such as Linux, Apache, Sendmail, or Jabber. These publicly and freely available software packages have reached wide diffusion as they are of high quality, despite the fact that they were, at least initially, not supported by any commercial company. Rather, they grew out of geographically dispersed communities of developers collaborating over the internet. The process these communities use to develop OSS – the so-called 'open source process' – contradicts most textbook-knowledge on software engineering, but proved very successful. Its power derives from the openness and public availability of the code, which allows any interested programmer to use, inspect and improve the code. Furthermore, comments, error corrections ('bug fixes') and additional code can be sent to the maintainer of the OSS project at very low cost."* [43, p. 360].

A theoretical explanation of the success of open source is given by von Hippel and von Krogh[44]. Whereas the private sector (commercial R+D, software development) creates innovation by investing in development and withholding intellectual property rights in order to be able to able to exploit their innovations commercially, the state funds collectively funds research through universities, where the expectation is that new knowledge created through innovation will be made publically available. Open source operates a new private (not supported by the state) collective (open access) model.

The attraction of open source for software developers is that it shortcuts the development process; here are very many programs with many different functionalities already running and debugged. Why should you duplicate writing this code if you can get it for free? This can speed up time to market and allow you to get some products released and thus generate income. A second attraction is ease of integration with other open source software. There are also drawbacks - it's not always easy working with code you've not written yourself, and some types of OS licence stipulate that you also release your own code.

Software value

At the centre of every software venture is the generation of value, most commonly understood as a profit in a commercial venture. Because open sources initiatives involve both working for free, and giving code away, it's important to examine where the value of software lies. Software is generally understood to be a form of intellectual property: that is that writing software is a creative form of encoding knowledge, both technical (about how computer hardware and interacting software work) and situational (about the practice of the user and what can add value for them). Therefore the result is intellectual property to which the rights can be protected by law. Software is exceptionally

easy to copy, and is widely copied, so without the protection of the law it would be difficult to continue to sell a software product. Various forms of protection are available, without much international standardisation, so the area is a complex minefield:

- patent
- copyright
- trademark
- industrial design right
- trade secret
- licensing
- non-disclosure agreement

The most traditional model for working with software value is practised by product-oriented companies such as Microsoft and Apple - you develop a product which is protected by copyright and you then sell or licences that product. You hold the intellectual property as secret as possible to avoid you competitors duplicating it and thereby eroding your market. A consultancy software company sells its development services and makes closed source software for a client, who often then owns the copyright. An extension of this principle is the patent system, where innovative and non-obvious technical advances can obtain further legal protection. These are typically granted for the mathematical or technical principle behind the software (for instance a new encryption algorithm), rather than the software itself. Thus, patents protect general-purpose technologies which can be used in a number of different products (Giarratana). The closed source or proprietary strategy, which exploits this protection) has dominated the market for many years. However there are many signs that this model has partially broken down in the last fifteen years. Internet routing is dominated by open source software, Linux is a competitor in the operating systems market, internet service companies such Google make software which is free to use and make money by other means. There are a variety of value proposition strategies open to It professionals and software entrepreneurs, where the traditional strategies are only part of the picture. The traditional value propositions are also heavily populated by established companies, implying that an alternative value proposition can be a way to break into an existing market. This is what Google do when they offer calendar, spreadsheet and word-processing software as free internet services.

It follows that software entrepreneurs need to understand the relative advantages of protecting their intellectual property and of sharing it through the various General Public and Creative Commons licences, and various different income generation strategies which are consequent upon these choices. Giarratana[24] for example shows how patent protection in the encryption industry led to two income sources: from off-the-shelf software products and from licensing the patents to other software companies.

Open business models

Chesbrough's account of open innovation (using openness principles to generate innovation) and open business models (generating value by sharing parts of your intellectual property) help us to understand a wider variety of value generation strategies. He contrasts closed and open innovation principles like this[45]:

closed innovation principles	**open innovation principles**
the smart people in our field work for us	not all the smart people work for us - we need to work with smart people inside and outside our company
to profit from R+D. we must discover it, develop it, and ship it ourselves	external R+D can create significant value - internal R+D is needed to claim some portion of that value
if we discover it ourselves, we will get it to market first	we don't have to originate the research to profit from it
the company that gets an innovation to market first will win	building a better business model is better than getting to market first
if we create the most and the best ideas in the industry, we will win	if we make the best use of internal and external ideas, we will win
we should control our IP, so that our competitors don't profit from our ideas	we should profit from others' use of our IP, and we should buy others' IP wherever it advances our own business model

If we translate these principles into closed and open value generation for software companies we get:

closed (proprietary)	**open and mixed source**
we have the ideas ourselves	we develop some of the ideas with others
we write the code ourselves	we write those parts of the code which add value
we sell the code	we give away some of our code, and have other revenue generation strategies
we protect our intellectual capital	we share some of our intellectual property

Open source companies employ these open and mixed business models. Often they provide both a free and a revenue generating version of what is essentially the same software, with a variety of different value-adding services.

company/organisation	**open source/free**	**revenue generation**
Canonical	Ubuntu	Technical support
Red Hat	Fedora (project)	Red Hat Enterprise Linux (RHEL)
Novell	openSUSE (project)	SUSE Linux Enterprise (SLE)
Sun Microsystems	OpenOffice.org	StarOffice
Adobe	Flex	Flash Builder IDE
Apple	Darwin	Mac OS X
Francisco Burzi	PHP-Nuke v.n-1	PHP-Nuke v.n
Ingres	Ingres database	Subscription for service and support, Ingres Icebreaker Appliance database
MySQL	MySQL	MySQL enterprise (subscription, support and additional features)
Linspire Inc.	Freespire	Linspire
Mandriva	Mandriva Linux, Mandriva Linux One	Mandriva Linux 2008
Mozilla Foundation	Firefox	Google

There are various advantages in the open strategies. There is a value in community (for example shared knowledge, better feedback processes and dissemination, and free marketing) which are also typical in open source communities. Integrated user communities offer domain knowledge, design input, feedback, innovation and future work. Colleague developers share code, provide quality assurance (for example de-bugging), exchange knowledge and develop future co-operations. IT and software firms with complementary expertise co-operate on projects to reduce development costs. Software innovators co-operate with researchers to have access to leading edge technical advances and funding though research projects. Software entrepreneurs co-operate with business people to improve their management skills and win investment.

Collabra

Collabra developed collaborative work tools which competed in the 1990's with the dominant product Lotus Notes. The company was started in stealth mode because they were afraid that Lotus would use its dominant market position to put them out of business. They were, however, relatively open, revealing many of their code secrets to their customers and third-party software developers, under non-disclosure agreements (totalling 195). This helped them to build a significant knowledge and code base, together with a good reputation in the business. These were attractive to Microsoft who lagged behind in this software type and need to compete with Lotus. They made a partnering agreement and developed joint marketing strategies with them which allowed them to achieve a significant share of the market. They were eventually sold to Netscape for $107m.[3]

However the classic dangers of sharing intellectual property to not disappear with open business strategies, and some degree of care is needed to avoid putting yourself at a disadvantage with larger and much more powerful competitors in the market.

Go Corporation

Go Corporation developed PenPoint, an operating system for pen computing, they went through six years of development with several hundred developers, and $750m venture capital. They were careful to protect their intellectual property, operating a commercial secrecy policy, and obtained several patents. They shared some of their technical knowledge with Bill Gates, being careful to get him to sign a non-disclosure agreement, designed to protect their IP. They were later acquired by AT+T where PenPoint ran on some personal communicators (none of which were a commercial success) but their operating system was completely eclipsed by Microsoft's PenWindows. PenWindows was later the subject of a Federal Trade Commission investigation and patent violation suits by GO, but these were not upheld. The company was divested and disbanded by AT+T.

Chesbrough[3] classifies open source business models as follows

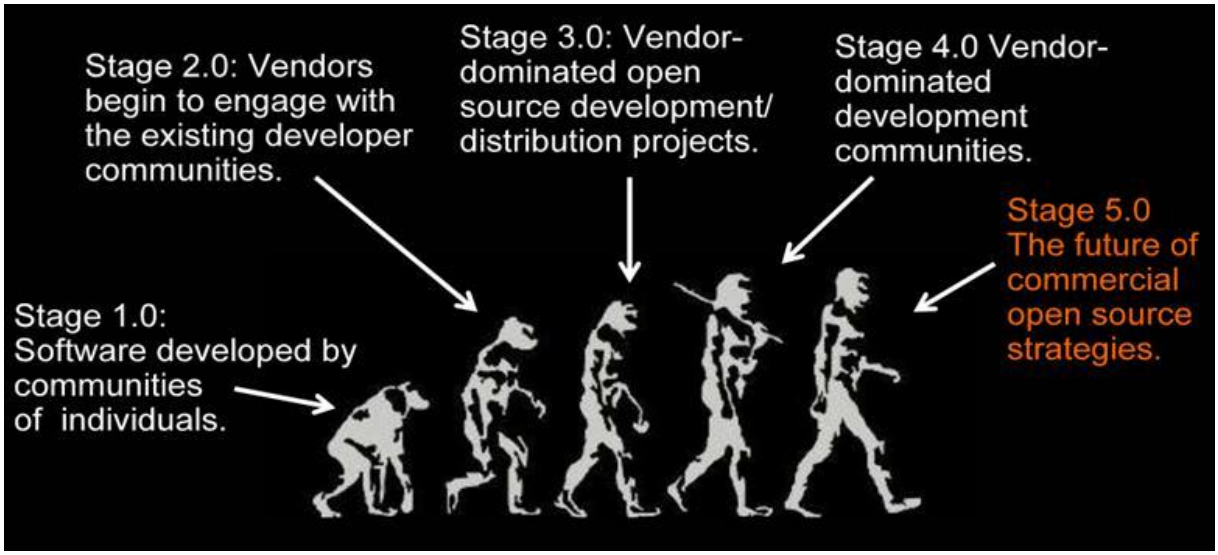
- selling installation, service and support (Red Hat)
- versioning the software with free version as entry level (MySQL)
- integrating software with other parts of a customer's IT infrastructure (IBM with Linux and Java)
- providing proprietary complements – building software based on, or in conjunction with the open source code.

Red Hat

Red Hat is an S+P 500 (major stock market index) company in the free and open source software sector, and a major Linux distribution vendor. Founded in 1993, Red Hat has its corporate headquarters in Raleigh, North Carolina with satellite offices worldwide. Red Hat has become associated to a large extent with its enterprise operating system Red Hat Enterprise Linux and with the acquisition of open-source enterprise middleware vendor JBoss. Red Hat provides operating-system platforms along with middleware, applications, and management products, as well as support, training, and consulting services. Red Hat creates, maintains, and contributes to many free software projects and has also acquired several proprietary software packages and released their source code under mostly GNU/GPL, while holding copyright under single commercial entity and selling looser licenses. As of February 2009, Red Hat was the largest corporate contributor to the Linux kernel. Red Hat sponsors the Fedora Project, a community-supported open-source project which aims to promote the rapid progress of free and open-source software and content. Fedora aims for rapid innovation using open processes and public forums. The Fedora Project Board, which comprises community leaders and representatives of Red Hat, leads the project and steers the direction of the project and of Fedora, the Linux distribution it develops. Red Hat employees work with the code alongside community members; many innovations within the Fedora Project make their way into new releases of Red Hat Enterprise Linux. Red Hat partly operates on a professional open-source business model based on open code, development within a community, professional quality assurance, and subscription-based customer support. They produce open-source code, so more programmers can make further adaptations and improvements. Red Hat sells subscriptions for the support, training, and integration services that help customers in using open-source software. Customers pay one set price for unlimited access to services such as Red Hat Network and up to 24/7 support. (Wikipedia)

Working with open source

The 451 group of consultants[46] argue that open source is not in itself a business model but 'development and distribution model that is enabled by a licensing tactic.' As such it can be part of a business strategy. They identify five stages in the evolution of these strategies.



Although the widely held image of open source is of dedicated independent developers fighting a valiant battle against big software corporations to protect the ideal of free software, this is no longer the case. Vendors, often the big software corporations, have understood the benefits of open source and involved themselves to the point where many developments are vendor-dominated. The elements of an open source business strategy[46] follow here:

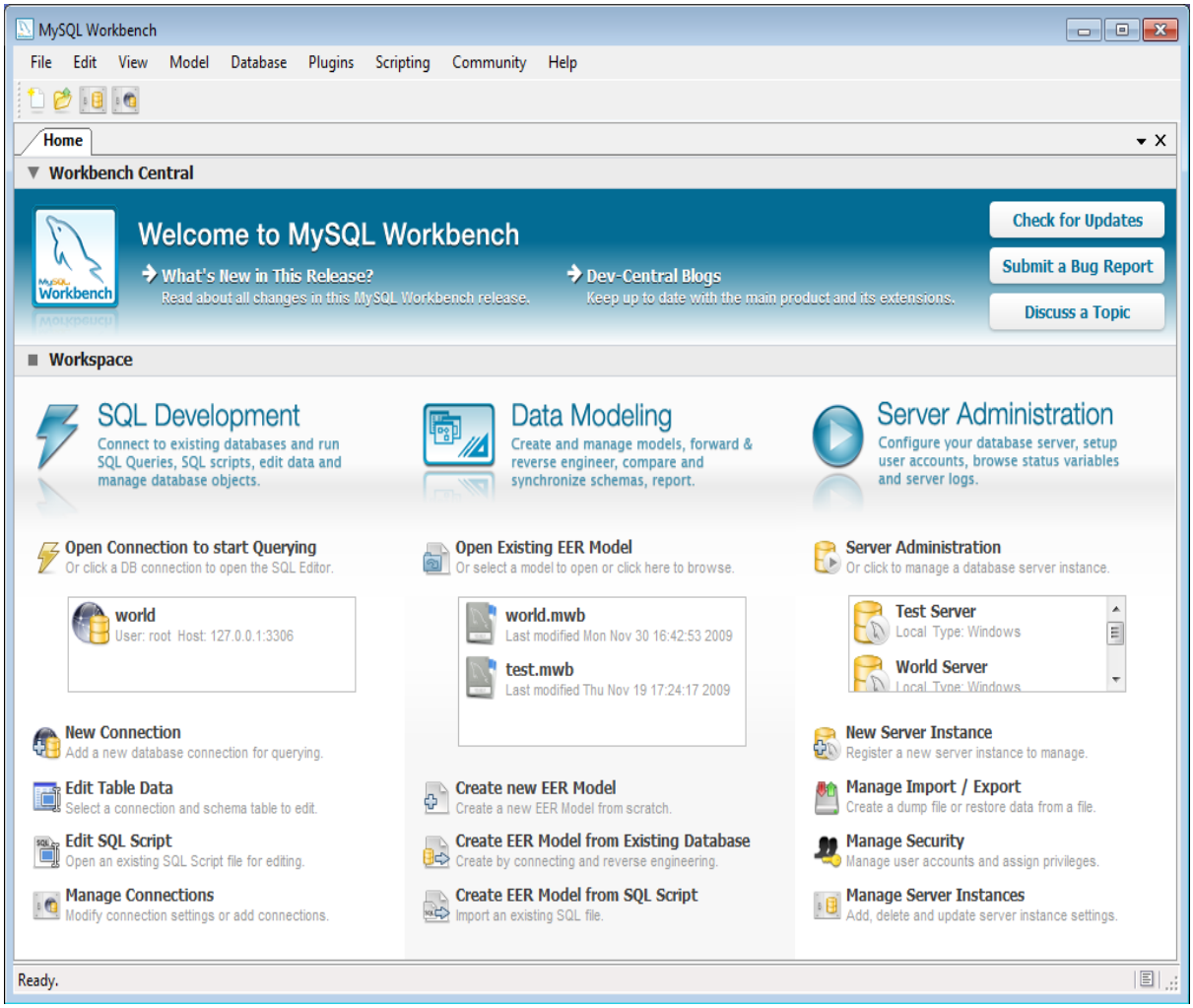
software license (which type of open source license is used)	reciprocal (requires that modifications must be licensed using the same license as the original)
	permissive (enables redistribution under a small set of rules)
development model (how a vendor collaborates with the community to produce code)	vendor open source - the software is distributed using an open source license; all contributions are public, development dominated by vendor developers
	community open source - distributed using an open source license, developed in public by a community of individuals and/or vendors
	mixed open source - based on a combination of projects using open source software developed publicly by multiple vendors and/or communities
	hybrid - underlying project distributed using an open source license, some functionality developed behind closed doors
vendor licensing strategy (how the software is licensed to the end-user)	dual licensing - single code base is licensed to different users using either an open source or a commercial license
	open core - core code available through open source license, enterprise or professional versions include open source code and closed source extensions, licensed commercially
	open-and-closed - open source products complemented by separate closed source products, developed and sold separately under a commercial license
	single open source - single code base with a single, open source license
	assembled open source - code from multiple open source projects
	closed - based on open source code but not available under an open source license
revenue triggers (how income is generated)	commercial license
	subscription
	service/support
	embedded hardware - open source software distributed as part of a hardware appliance
	embedded software - open source software embedded within larger commercial software package
	software as a service (SaaS) - users pay to access the software via the internet
	advertising - the software is free and funded by associated advertising
	custom development - customers pay for the software to be customized to meet their requirements
	other products and services - open source software not used to directly generate revenue, complementary products provide

Software entrepreneurs should choose complementary development, licensing and revenue strategies in order to maximize revenue-generation opportunities.

> *'it is easy to understand why vendors believe a combination of open source and proprietary models can provide the best of both worlds. The open source development and distribution models can increase the quality of the code developed, lower development and marketing costs, and increase opportunities for vendors to attract new potential customers. Meanwhile, commercial licensing is the tried-and-true method that software vendors use to build a commercial relationship with a customer. Most vendors generating revenue from open source software are trying to balance the benefits of open source development and commercial licensing.'* [46]

MySQL

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, MyBB, phpBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google (though not for searches), Facebook, and Twitter. As of April 2009, MySQL offered MySQL 5.1 in two different variants: the open source MySQL Community Server and the commercial Enterprise Server. MySQL 5.5 is offered under the same licences. They have a common code base.



MySQL's business strategy looks like this:

software license	**reciprocal**
development model	vendor open source
vendor licensing strategy	dual licensing
revenue triggers	commercial license
	subscription
	service/support

Open source and the software entrepreneur

Open source helps with liabilities of newness (few resources, no reputation, poorly established engineering practice, expertise gaps) and smallness (few developers, long time to market). It is therefore a possible solution to the challenges of market entry for start-ups. According to Gruber and Henkel[43], participation in OS projects contributed to embedded Linux software start-ups through:

- reputation-building (as additions to the project are accepted) and networking (connections with other developers and potential customers)
- OS projects act as marketing channel for the new companies
- acceptance of code by the community is a stamp of quality

- OS allows them to take advantage of previously written code (e.g. device drivers) and contributes to development speed and code quality
- gives access (entry wedge) to markets dominated by large players (e.g. embedded operating systems) where a small company cannot reproduce the necessary development effort to compete
- code can be extensively tested and improved by other participants
- the costs of switching for customers are reduced - it runs on Linux which is free and the customer is not locked in to expensive version updates and service as they are with Apple or Microsoft.
- the cost of entering the market is reduced.

'I once published an article in a relatively high ranking publication……. addressing higher management …….that was relatively expensive and yielded exactly zero responses. My job is not only to read mailing lists every day, but to respond also to really stupid beginner's questions ……. These are the things by which I have massively acquired new projects.' (embedded Linux software firm entrepreneur EU[43])

'It's hard to write device drivers. It's hard to write good application programs…….the more you can take advantage of ones that already exist, the more quickly you can get to focusing on what differentiates your product or your service from everyone else's product and their service'

'I know that our version of Linux is going to be extensively tested with both MySQL and with Oracle……..that would be something far beyond our ability ……. it would be an enormous financial burden for a company like QNX [vendor of proprietary embedded operating system]……but I get it for nothing, or for my participation in Linux.' (embedded Linux software company, USA[43])

Sources

ASLETT, M. (2008) Open Source is not a Business Model. the 451 Group.

GRUBER, M. & HENKEL, J. (2006) New ventures based on open innovation - an empirical analysis of start-up firms in embedded Linux. International Journal of Technology Management, 33, 356-372.

CHESBROUGH, H. (2003) Open innovation: The new imperative for creating and profiting from technology, Harvard Business Press.

CHESBROUGH, H. (2007) Open business models: How to thrive in the new innovation landscape, Harvard Business Press.

LERNER, J. & TIROLE, J. (2002) Some simple economics of open source. The journal of industrial economics, 50, 197-234.

WEST, J. (2003) How open is open enough?: Melding proprietary and open source platform strategies. Research Policy, 32, 1259-1285.

Intrapreneurship

Intrapreneurship involves effecting change from within a company: promoting a product idea, a new work practice, or a move into working with new technologies. In the entrepreneurship literature a variety of topics are discussed under this and related headings including

- organisational transformation or renewal (very important in the software field where technologies can change at breakneck pace)
- frame-breaking or discontinuous change (radical changes which involve many upheavals)
- corporate entrepreneurship (leading to new business ventures, the development of new products, services or processes and the renewal of strategies and competitive positions)
- corporate venturing (investment in other companies with complementary goals)
- spin off (initiatives leading to the formation of new companies, with or without the support of the parent company)
- spin in (incorporation of external companies)

'Engineers', argues Menzel

> *'take up a strong position in innovation activities since they contribute to an important role in the creation, development and generation of new (technical) knowledge. Their technical expertise and skills are an important source for new technical ideas that might turn into new entrepreneurial opportunities. Engineers can be seen as a mixture of builders, adventures and problem-solvers and their objectives is to create technical artefacts and provide services to society[47]'*

The ability to change, innovate and develop has been associated with economic success, both at the firm level and at the level of society since Schumpeter first pointed out the connections. Menzel argues that 'entrepreneurship within existing organizations and its role in organizational renewal, innovation, and the creation of new businesses …… [is] a subject of interest because of its effect on revitalization and performance of the firm'[47]. However organisational change is not necessarily easy to achieve and the change theories we investigated earlier point out why. Organisations are complex, made up of individuals with opinions and feelings, political, and necessarily bound into some degree of traditional practice in order to make things work reliably. A larger software organisation may develop tendencies such as these: standard development procedures and quality standards are enforced to reduce mistakes; projects are managed for efficiency and return on investment, large projects are plan-driven and controlled against the plan; management avoids moves (such as learning a new programming technology) that risk the base competences and business and protects these at all costs, new projects grow mainly out of prior experience; developers are compensated uniformly according to the company schemes and promoted if they fit in well. In these situations 'innovations just do not happen unless someone takes on the intrapreneurial role[48]'.

When innovation fails: how IBM missed the personal computer explosion

In the late 1970's, 8IBM, completely dominant in the mainframe market, were watching the growing home computer market. Though they firmly believed that the future lay in mainframe computing, they decided to enter it. Their first attempt, the IBM 5100, was a dismal failure. They considered

buying the fledgling game company Atari, but decided to develop their own personal computer line, with a new operating system. The secret plans were referred to as Project Chess and the code name for the new computer was Acorn (not so convincing a nickname – they might have gone for apple). Twelve engineers, led by William C. Lowe, assembled in Boca Raton, Florida, to design and build the Acorn. On August 12, 1981, IBM released their new computer, re-named the IBM PC. The PC stood for personal computer, making IBM responsible for popularizing the term PC. They made the architecture open and it quickly became the international standard.

However, the previous year (1980), IBM approached Bill Gates to discuss the state of home computers and what Microsoft products could do for IBM's project. Gates suggested that Basic should be written into the ROM chip. Microsoft had already produced several versions of Basic for different computer system beginning with the Altair, so Gates was more than happy to write a version for IBM. IBM then decided to outsource the OS for the new IBM PC and asked Gates. Since Microsoft had never written an operating system before, Gates first suggested that IBM investigate CP/M (Control Program for Microcomputers), written by Gary Kildall of Digital Research. Kildall, however, refused to sign a non-disclosure agreement and IBM gave Microsoft the contract to write the new operating system, MS-DOS. Gates based the new OS on QDOS, the "Quick and Dirty Operating System" written by Tim Paterson of Seattle Computer Products, buying the rights for $50,000 (whilst keeping the IBM/Microsoft deal a secret). Bill Gates then talked IBM into letting Microsoft retain the rights to market MS-DOS separately. The rest, as they say, is history. With an open architecture and Microsoft owning the rights to MS-DOS, the PC market eventually took off, but with Microsoft, not IBM as the big winner.

You, the intrapreneur, may have some special psychological characteristics (vision and creativity, initiative, internal motivation, autonomy, risk taking, internal control, commitment and persistence, market knowledge and customer orientation, knowledge of organizational structures and willingness to cross functional borders[47]); it's not clear that these amount to much more than being competent, experienced and having your own ideas. However your individual intrapreneurial behaviour may be somewhat different from standard organisation behaviours[20].

typical intrapreneurial behaviours	standard organizational behaviours
exploration of new business opportunities	exploitation of existing business activities
deflection from the present practice	reinforcement of the present practice
revolutionary change	evolutionary change
uncertainty acceptance	uncertainty avoidance
long-term (future) orientation	short-term orientation
flexibility, room to manoeuvre	planning and formalization of activities
visionary and intuitive decision-making	decision -making influenced by politics
holistic approach	functional expertise
fair compensation depending on venture success	traditional compensation independent of venture success

Let us say you are experienced in agile development, you move to a company with a more traditional development style and you want you introduce some agile practices. This is included under my definition of 'promoting a software venture.' If you are the managing director you at least have the authority to decide this; if you are less senior you have an intrapreneurial task to achieve. It's possible that you have some good arguments and will be listened to, but it's more likely that

your organisation has good reasons for its traditions and is quite resistant to change. We'll investigate two schools of thought about how you can achieve this: one based on some rational prescriptions from the theorist, and one based on some harder lessons from practice.

Rational prescriptions (associated with ADE)

In this version of intrapreneurship, good innovative ideas eventually win through, but the need some determination, perseverance and skill from the intrapreneur and the correct organisational conditions (which are assumed to be set by senior managers). According to Kotter, you should:

1. establish a sense of urgency (point out how the company's development style leads to poor, overpriced software and unhappy customers)
2. create the guiding coalition (align yourself with others that believe the same)
3. develop a vision (articulate how projects will run more effectively with an agile method)
4. communicate the change vision (explain and practise agile ways)
5. empower broad-based action (partner with a senior project manager who agrees with you)
6. generate short-term wins (find a small and easy project where you can experiment with the agile method of your choice)
7. consolidate gains and produce more change (move on to some larger and more complex projects)
8. anchor new approaches in the culture (gain senior management approval and alter the company's standard procedures).

This plan enables you to operate some features of the Analyse, Design, Enact (ADE) paradigm.

However, if organisational conditions are against you (your IT company is characterised by tradition, history, vested interests, authority patterns, internal politics, common practice, conservatism and risk avoidance), you may still fail. Barriers to innovation may include overt non-constructive criticism of new ideas, lack of team work in new product development, the practice of copying competitors ideas, and poor links with knowledge institutions (such as universities). Therefore managers of engineering companies have a special responsibility to maintain conditions where intrapreneurs can flourish. Menzel[47] argues that five factors are particularly important:

- a physical environment which stimulates creativity, allowing spaces both for private reflection and different types of interactions (e.g. Microsoft where corporate headquarters resemble a college campus with high social integration and dynamics)
- reduction of hierarchy and bureaucracy (flat structures without too much hierarchical control and the avoidance of too much routine work)
- top management encouragement ('it is top management's task to communicate and fill with life the organization's vision, goals, and strategy`…… based on clear commitment to intrapreneurship initiatives……..a strategy of innovation …….. setting goals for innovators to achieve ……..leave the people free to innovate…….. use the goals and values of the organization to guide behaviour, not rules, procedures or reward and punishment[47],
- advocates/champions who are prepared to support intrapreneurs
- resources including time (often a percentage of the engineer's work time) and access to capital including 'patient' money which is not withdrawn at the first sign of a problem

Kuratko and Hodgetts[20] summarize the problems inherent in established companies, the adverse effects these create on intrapreneurship and recommended actions for responsible managers:

traditional management practices	adverse effects	*recommended actions*
enforce standard procedures to avoid mistakes	innovative solutions blocked, funds misspent	*make ground rules specific to each situation*
manage resources for efficiency and return on investment	competitive lead lost, low market penetration	*focus effort on critical issues (e.g. market share)*
control against plan	facts ignored that should replace assumptions	*change plan to reflect new learning*
plan for the long term	nonviable goals locked in, high failure costs	*envision a goal, then set interim milestones, reassess after each*
manage functionally	entrepreneur failure and/or venture failure	*support entrepreneur with managerial and multidiscipline skills*
avoid moves that risk the base business	missed opportunities	*take small steps, build out from strengths*
protect the base business at all costs	venturing dumped when base business is threatened	*make venturing mainstream, take affordable risks*
judge new steps from prior experience	wrong decisions about competition and markets	*use learning strategies, test assumptions*
compensate uniformly	low motivation and inefficient operations	*balance risk and reward, employ special compensation*
promote compatible individuals	loss of innovators	*accommodate 'boat rockers' and 'doers'*

Intrapreneurship as independent action: an empirical view associated with CDA

Another way of characterising intrapreneurship, assuming that organisational conditions are not always perfect for innovation, is as independent, iterative and unconventional action. Abetti's case study at Toshiba[49] shows an example of intrapreneurial subterfuge, where much of the action (developing the company's highly successful PC's and laptops) is conducted 'under the table' (underground) though skunk works and a variety of actions hidden from the senior managers who repeatedly veto the project.

Laptop intrapreneurship at Toshiba

In the 1980's and 90's Toshiba developed highly successful lines of PC's and became one of the leading manufacturers of high quality laptops; however this was in spite of, rather than because of the strategies of senior managers. The success was attributed to the intrapreneurial actions of a senior engineer, Mizoguchi and his champions Koga and Nishida. Here is the story, summarised in nine phases by Abetti[49]

Phase 1 - latency. Toshiba, primarily a manufacturer of electrical equipment struggled to enter the mainframe market but could not compete with IBM. In 1978 they withdrew but kept their computer engineer group to preserve the core competencies.

Phase 2 - stillbirth. Mizoguchi developed the first Japanese PC but headquarters vetoed its commercialisation in Japan. Some collaborations in the US were launched, but failed.

Phase 3 - second stillbirth. In 1979-81 NEC succeeded with the first Japanese PC. Mizoguchi obtained permission to develop Pasopia 7 (Toshiba's response) but this failed miserably both in the Japanese and US markets. It arrived too late and was not IBM compatible.

Phase 4 - conception. In 1983 Mizoguchi with five teams of engineers visited the US and conceived a portable fully IBM-compatible PC (using the well-known 'back to the future' design approach, another innovation).

Phase 5 - gestation. All requests for development funds and transfer of staff were denied by headquarters and Mizoguchi, protected by Koga, stared a covert 'under the table' laptop project, siphoning funds and personnel from other projects.

Phase 6 - birth of laptop. Mizoguchi built seven prototypes one of which was seen by the VP of marketing in Europe. He sold 14,000 in fourteen months and the laptop won the 'king of laptops' award. Despite this, headquarters refused to commit funds for attacking the US market. Nishida instead used the European profits to do this.

Phase 7 - adolescence. In 1987-88 Mizoguchi's team was re-incorporated into the mainstream business in a new unit managed by Mizushima. They developed a Japanese version of the laptop and a corporate committee successfully marketed it

Phase 8 - birth of notebook. Many competitors emerged in the laptop market and Toshiba lost market share. Mizoguchi (under the table) developed a smaller, lighter notebook.

Phase 9 - adulthood. Between 1990 and 1996 the contributions of Mizoguchi, Koga and Nishidi were acknowledged with promotions - (Koga to senior VP and member of the board). Development of subnotebook, palmtop and multimedia PC's was established in a corporate project called Advanced 1.

The intrapreneur therefore can have something of a rebel role in relation to the organisation they work in. According to Harrison[38], the role of the corporate entrepreneur to 'corral resources, steal personnel time, conceal development activities, and curry personal favours to secure the resources needed for their new ventures.' Pinchot[48] offered ten commandments for the effective intrapreneur:

1. come to work each day willing to be fired
2. circumvent any orders aimed at stopping your dream
3. do any job needed to make your project work, regardless of your job description
4. network with good people to assist you
5. build a spirited team; choose and work only with the best
6. work underground for as long as you can - publicity triggers the corporate immune mechanism

7. be loyal and truthful to your sponsors
8. remember it is easier to ask forgiveness than permission
9. be true to your goals, but be realistic about the ways to achieve them
10. keep the vision strong

The role of champion in relation to the new venture is important, especially in larger organisations. Changing things in a large machine where you are a small cog is difficult and you are often reliant on the support of a senior colleague who believes in you and will protect you and help to spread the message. Day[50] offered three hypotheses:

- the lower the principal champion's hierarchical level, the more innovative the venture will be.
- in general, principal champions from corporate headquarters, particularly from staff positions, will be negatively associated with innovativeness.
- as retroactive legitimizers, top managers enforce only those ventures that are proven successes, and then only after they have established themselves as such.

We should conclude that, in many situations, top managers are not necessarily the source of much innovation and need to be coaxed into co-operation. Abetti[49] suggests that four kinds of championing phases are critical if new venture ideas are to survive: idea generating, opportunistic behaviour, resource gathering and incorporation into the mainstream business.

phase	champion	role
ideas	Mizoguchi	develop back to the future approach and gain acceptance by engineering team
opportunistic behaviour	Koga	continuously probe markets in the US and Europe; condone rule bending and protect team from interference by headquarters
resources	Koga	maintain profitability of computer business and divert resources to entrepreneurial team; keep the team small to conserve resources; enlist the support of Europe and US marketing
incorporation	Mizushima	obtain corporate acceptance of the new venture; legitimize the new venture as part of mainstream; marshal extraordinary corporate resources for rapid growth

If we return to your problem of introducing some agile development practices into a traditional company, then we might now understand that the rational prescriptions offered by Kotter and others may not always work in every situation. A more flexible, adaptive, improvisatory and sometimes subversive approach may be better; this we can align with Consider, Do, Adjust (CDA).

Where an intrapreneurial venture is successful it is rather common for others (often senior managers) to take the credit, but many, maybe most, fail. What are the options if your venture is intimately rejected? You may leave the company and pursue your venture, perhaps in the form of a start-up. You may display loyalty to your employers and swallow your disappointment. You can complain loudly, or you can resort to subterfuge.

Sources

ABETTI, P. A. (1997) The birth and growth of Toshiba's laptop and notebook computers: A case study in Japanese corporate venturing. Journal of Business Venturing, 12, 507-529.

DAY, D. (1994) Raising radicals: Different processes for championing innovative corporate ventures. Organization Science, 5, 148-172.

HARRISON, R., MASON, C. & GIRLING, P. (2004) Financial bootstrapping and venture development in the software industry. Entrepreneurship & Regional Development, 16, 307-333.

KURATKO, D. & HODGETTS, R. (2004) Entrepreneurship: Theory, Process and Practice, Mason, Ohio, Thomson.

MENZEL, H. C., AALTIO, I. & ULIJN, J. M. (2007) On the way to creativity: Engineers as intrapreneurs in organizations. *Technovation,* 27**,** 732-743.

PINCHOT, G. (1985) Intrapreneuring: why you don't have to leave the corporation to become an entrepreneur, Harper & Row New York, NY.

WEST, J. (2003) How open is open enough?: Melding proprietary and open source platform strategies. Research Policy, 32, 1259-1285.

Conclusion: motivational questions and answers; two paradigms and four themes revisited

What forms of value can software create?

Software should be understood as the translation of various forms of related knowledge into code; this is a complex intellectual activity so software should be understood as a form of intellectual property. You can't unfortunately pay your developers in IP or feed your family with it, so IP has to be further translated into economic rewards: revenues which hopefully outstrip costs. This second translation is the task of a business model; these are many, varied, sometimes complex, and a software firm can operate many business models simultaneously - building its total revenue in different ways. A software start-up will need explicit thinking about these translations, even if it doesn't choose to call that thinking by the name 'business model'.

How do you recognise your intellectual property in your software work and protect it?

Protection for intellectual capital is built into the legal structure of modern market economies, as patent, copyright, trademark and so on. Software is easily copied so these protections are necessary to prevent theft. A historical way of thinking jealously guards intellectual property: proprietary software is built, the source code is completely closed, and licenses are sold. However the modern industry has also learned how to make money by giving away the intellectual property that software represents. In between the closed and the open models is every degree of shading (for example over seventy different kinds of reciprocal and permissive OS licences). Many strategies are possible; the historical style is not necessarily the best when trying to enter the modern market.

How do you leverage open source software and the open source development model?

Open source provides (sometimes) free code: operating systems, device drivers, middleware, network software, various standard applications, business software. Most young companies don't have the resources to build these themselves and if they did they would not be able to easily derive revenues from them. There are also network and reputation advantages. On the debit side, it's a more complex approach to building IP and translating it into revenue. The various open source business strategies are beginning to be better understood (judicious combinations of revenue triggers, development strategies, licensing options) but there is always room for innovation.

How do you make a business out of a software product idea?

Many It professionals start a new company because they know they have a good idea for an innovative software product; sometimes they have developed the science behind the software; sometimes they also have a prototype. The paths to developing a business out of the idea are many and varied, and there are many issues to consider. In one strategy the innovators can develop a fully-fledged business plan and seek venture capital to put it into practice (Analyse, Design, Enact). Ina second they can bootstrap their business, looking for collaborators with commitment, writing software as they able and being very flexible in responding to the opportunities they recognise (Consider, Do, Adjust). There are many variations and you can start in CDA mode and become more analytical and design oriented later in the search for capital to stimulate business growth. IT and software professionals have the most important skills (writing software) for these businesses, but there are a variety of other business skills and competences that must be learned or acquired. However we don't agree with the opinion often expressed in business literatures that business skills

are the really important ones. Look at the great entrepreneurs in software and software-intensive businesses: they were trained to write code, even if they don't do it anymore.

What business models do young software and software-dependent firms use?

The most basic distinction is between companies offering finished software products and companies providing various kinds of software services (the most important of which is writing tailor-made software for clients). A more detailed classification is:

- software tailoring – building tailor-made software for specific customers
- applied formats – customized solutions based on common platform
- resource provisioning – developing software components or middleware designed to integrate with other software
- standard offerings – own products sold widely and used without customization

in addition to these there are a variety of open business models, generic models for (non-software)businesses and a variety of generic models for e and m commerce. However it's not really a question of copying an existing model; more of understanding the various different models, tailoring, adapting and innovating. There are as many successful software business models as there are successful software firms.

How do you commercialize and market a new software product?

We didn't investigate this question in depth, but one answer is: in collaboration. The first customer is often very important for refining the first product, and other better established firms as partners help with marketing and distribution problems, as well as internationalisation and breaking into new markets.

How do you understand the market you are in, or create a market if it does not exist?

The strength of the ADE paradigm is analysis; there are a variety of conventional business models (for example Porter's model of competitive forces) that will help you to understand your business environment and find the niche that you want to occupy, or the opportunity that other software competitors have not yet exploited. If you are more confident in your business idea, more action-oriented (or perhaps a genuine visionary) then you will assume that your product or service is innovative and that no market exists just now, but that it can be created by your actions. This is an assumption of Sarasvathy's theory of effectuation. Now you must build a market, customer by customer, adapting and learning as you go.

How do you find the necessary resources to develop your firm or product idea?

Venture capital, or an angel is the ADE answer. Bootstrapping is the CDA alternative, and also the most common way for young software entrepreneurs. Partnership and mutual knowledge sharing provide the necessary intellectual resources. But what resources do you really need? the most important resource (the ability to write effective code) you have already. Get started and at the appropriate time sorry about growth.

Who becomes a software entrepreneur, and what skills and capabilities do you need?

At one time researchers believed that entrepreneurs had particular character traits which made them different from other competent professionals (for example software engineers and managers working in companies). This is not really the case; a software entrepreneur is a competent IT

professional who has chosen to start a company. Perhaps more important is the match of relevant experiences, knowledge and the ability to learn fast and adapt.

How do you manage a start-up?

Young companies take time to develop standardised management practices, so the answer is flexibly, adaptively and in a learning mode. Ask for advice, outsource business functions that are not core where you have little expertise (accounting, marketing), and practice mutual engagement: very good and intense dialogue with your colleagues. If you're a real engineer type how primarily enjoys building software, consider recruiting someone with complementary business skills.

What engineering and development practices do you need to develop software in a very young firm?

The type of traditional software engineering practice which suits mature companies can't be achieved in accompany that is small, so the answer is informal agile practice. User/customer focus is also important for consultant-type software firms. Rigorous adherence to agile principles is also demanding, so small companies improvise their way to new products. However development practice rapidly becomes an issue in all software development work, so engineering practice also needs to develop quickly. Good practices with respect to testing, project management and a range of other issues need to be shared and incorporated in work routines.

How do you manage competitors and customers?

Product-oriented firms need to keep an eye on their competitors, adjusting their own products to remain innovative and competitive in relation to their competitors'. Sometimes there will be partnering and knowledge-sharing, but this need to be managed carefully. Service-style companies need to develop close relationships with their customers; understanding not only their software needs, but what drives their businesses and enterprise.

How do young software firms grow through networking and partnering?

Young firms suffer the disadvantages of smallness and newness and lack many kinds of resources, the most important of which is often knowledge. Partnering offers access to more development resources, previously developed code, other kinds of complementary skills and knowledge, and new financing sources. Partnering and networking often provide a wider customer-base, new markets, the chance to become international and better distribution channels.

What is the role of innovation in software entrepreneurship?

Innovation and entrepreneurship are closely linked in the IT world. Often a start-ups key to survival is the innovation in their first product, and their innovation speed: the ability to continue to innovate and to bring their products to market before their competitors. However innovation is not a necessary component for software entrepreneurship; a service-oriented company can also survive on good customer relations, efficiency, and cost leadership

How does a software start-up survive and grow?

Nobody really knows the answer to this question. If they did there would be many fewer failures. Many factors contribute, in sometimes unpredictable patterns.

Why do software ventures fail?

The shallow answer is that they run out of money; however the causes of this may be many. The innovative product wasn't so innovative, or a major competitor immediately imitated it. The market

collapsed as a result of a substitute product. A large competitor practised price-undercutting. There were technical difficulties and development delays, or arguments with customers and partners. The bank lost confidence. The founders ran out of steam and got stressed. A hundred other reasons, in any combination. Many start-ups fail, in the sense that the company ceases to exist. However the outcomes for the individual entrepreneur(s) are not necessarily bad unless they have been unusually naïve. The company is bought by a larger rival, or they merge with a partner. The founder moves on to a new start-up or is attractive to another company because of their management experience or specialised knowledge.

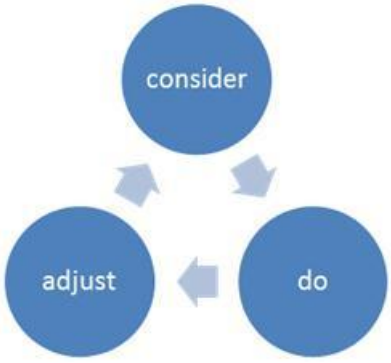
How do you promote a software product or practice from within a software development firm?

This is the important practice of intrapreneurship, without which every company (with the possible exception of Apple when run by Steve Jobs) will eventually die. There are some rational models for making this happen, but as often as not the pattern is adapt and survive in the face of managerial indifference or hostility.

Two paradigms for software entrepreneurship

We developed two paradigms for software entrepreneurship: Analyse, Design, Enact:

and Consider, Do, Adjust:



Here they are, summarised, again:

	analyse, design, enact	consider, do, adjust
theoretical inspiration:	standard entrepreneurship literature	Sarasvathy: theory of effectuation
software engineering inspiration:	traditional development	agile development
description:	promote a software venture by understanding the technical and	promote a software venture by taking iterative and incremental

	business environment, designing a logical response and setting it in motion	steps forward on the basis of what is achievable now
process:	sequential	iterative
understanding of future:	predictable, non-controllable – adjust to the forces of the market	unpredictable, controllable – create (a small part of) the future
attitude to market:	analyse the market and choose a position or niche (find opportunity)	avoid or outsmart obvious competitors and create a new market (make opportunity)
attitude to technology development:	understand technology trajectories and fit software projects into likely developments	develop the areas of technology expertise you excel in
role of business planning:	conceptualise the venture as completely as possible before starting	take initial business decisions based on what you can afford and look out for what you need to do next
software development style:	decide software engineering practice up-front, likely to emphasise the traditional	improvise engineering practice and decide according to circumstance, lean practice, agility
attitude to change:	avoid change as far as possible - stick to plan to achieve goals	embrace change as opportunity and act upon new situations and possibilities
funding approach:	attract capital and investors before start up - fund to maximize return	invest what you can afford without considering the possible return and bootstrap
approach to others working in the same areas:	avoid competition - consider competitors a threat to market position	collaborate with those who demonstrate commitment and network with potential stakeholders
approach to intellectual property:	protect IP through copyrights and patents to deter competition, improve market share	build networks of ideas, sharing, various business models including open source, mixed open and closed code access
partnering and networking	control collaboration to avoid losing intellectual property	build a network of committed stakeholders, collaborators and partners
time to market	long on the basis of venture capital	short to maximise returns

They serve as a pedagogical and analytical devise to understand the way researchers write about entrepreneurship, and to adapt these ideas to the software and IT arena. Next we will consider the four major themes (the software start-up, e(m)Entrepreneurship, open entrepreneurship and intrapreneurship) in the light of the two paradigms.

The software start-up

Much of the standard textbook literature promotes the ADE approach to starting a business, but this is largely because it reflects a heavily analysis-biased mode of thinking common to the scientists who

do the research and write the books. This bias is also reflected in the research articles we read. Case studies written by practitioners of practice-oriented commentators reveal a more mixed picture. There are many elements of CDA in (for instance) Kaplan's account of Go Corporation besides the glamorous venture capital rationalistic large scale approach. For many young software entrepreneurs Sarasvathy's model and CDA will be more appropriate, because resources for analysis and planning are not available, the likelihood of obtaining venture capital small, and events are too unstable and fast moving. Bootstrapping is an alternative means of raising limited amounts of capital, and affordable loss a useful principle.

E-entrepreneurship

The picture is the same as for software start-ups. Koller's research-oriented account promotes a ADE approach, whereas descriptive accounts by entrepreneurs (e.g. Omidyar) lean in another direction. There don't seem to be any intrinsic reasons why this kind of start-up should be different and the emerging market for mobile services and apps seems to be rich in opportunity for engineers without many resources employing the CDA approach.

Open entrepreneurship

Working with open source and open business models is a more modern and less conventional approach to software entrepreneurship. Most of the ADE way of thinking is based on historical traditions of protecting innovation and intellectual capital - this is built into the analysis tradition (analyse the past to predict the future). CDA is better positioned to work with fast changing types of value propositions where there is no stable precedence.

Intrapreneurship

The literature we learn some rational models for how to promote a new venture from inside an existing company, together with suggestions for what managers should do to organise a culture where intrapreneurship is accepted, since this is clearly beneficial for the company in the longer run. However empirical accounts describe a somewhat cynical view of organisations and organisational change, where the rational (ADE) approach is unlikely to be successful. However Sarasvathy's account of opportunism, experimentation and commitment building (CDA) matches the case study accounts from Toshiba rather well. This style of intrapreneurship is also supported in Pinchot’s ten commandments.

References

1. Cusumano, M., *The business of software: What every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad*. 2004, New York: Free Press.
2. Kaplan, J., *Start Up*. 1994, London: Warner.
3. Chesbrough, H., *Open business models: How to thrive in the new innovation landscape*. 2007: Harvard Business Press.
4. Osterwalder, A. and Y. Pigneur, *Business Model Generation*. 2010, Hoboken, New Jersey: Wiley.
5. Porter, M.E., *Competitive advantage: creating and sustaining superior performance*. 1985, New York: Free Press.
6. Barney, J.B., *The resource-based theory of the firm.* Organization Science, 1996. **7**: p. 469-469.
7. Dollinger, M.J., *Entrepreneurship: strategies and resources*. 1999, Upper Saddle River, New Jersey: Prentice hall.
8. Grant, R.M., *Toward a knowledge-based theory of the firm.* Strategic Management Journal, 1996. **17**: p. 109-122.
9. Prahalad, C. and G. Hamel, *The core competence of the corporation.* International Library of Critical Writings in Economics, 2003. **163**: p. 210-222.
10. Porter, M., *The five competitive forces that shape strategy.* Harvard Business Review, 2008. **86**(1): p. 78-86.
11. Kotter, J.P., *Leading change*. 1996, Boston: Harvard Business School Press.
12. Cougar, D., L. Higgins, and S. McIntyre. ``*Differentiating creativity, innovation, and entrepreneurship for information service products and processes*''. 1990.
13. Shumpeter, J.A., *The theory of economic development.* Cambridge, MA: Harvard Economic Studies, 1934.
14. Kirzner, I.M., *Competition and Entrepreneurship*. 1978, Chicago: University of Chicago Press.
15. Rose, J., *Software Innovation: eight work-style heuristics for creative software developers*. 2010, Aalborg: Software Innovation, Dept. of Computer Science, Aalborg University.
16. Rae, D., *Entrepreneurship: from Opportunity to Action*. 2007, New York: Palgrave.
17. Sarasvathy, S., *Effectuation: Elements of entrepreneurial expertise*. 2008: Edward Elgar Publishing.
18. Varis, J., O. Kuivalainen, and S. Saarenketo, *Partner selection for international marketing and distribution in corporate new ventures.* Journal of International Entrepreneurship, 2005. **3**(1): p. 19-36.
19. Kuratko, D.F. and R.M. Hodgetts, *Entrepreneurship: A contemporary approach*. 1995: Dryden Press.
20. Kuratko, D. and R. Hodgetts, *Entrepreneurship: Theory, Process and Practice*. 2004, Mason, Ohio: Thomson.
21. Read, S., et al., *Effectual Entrepreneurship*. 2011, London: Routledge.
22. Hsu, D.H., E.B. Roberts, and C.E. Eesley, *Entrepreneurs from technology-based universities: evidence from MIT.* Research Policy, 2007. **36**(5): p. 768-788.
23. Aaen, I. and J. Rose. *A Software Entrepreneurship Course: between two paradigms*. in *15th Annual Interdisciplinary Entrepreneurship Conference*. 2011. St. Gallen and Zurich.
24. Giarratana, M.S., *The birth of a new industry: entry by start-ups and the drivers of firm growth - The case of encryption software.* Research Policy, 2004. **33**(5): p. 787-806.
25. Igel, B. and N. Islam, *Strategies for service and market development of entrepreneurial software designing firms.* Technovation, 2001. **21**(3): p. 157-166.
26. Mueller, T. and H. Gemunden, *Founder team interaction, customer and competitor orientation in software ventures.* Management Research News, 2009. **32**(6): p. 539-554.
27. Rajala, R., M. Rossi, and V. Tuunainen. *A framework for analyzing software business models*. 2003: Citeseer.

28. Rajala, R. and M. Westerlund, *Business models a new perspective on firms' assets and capabilities: observations from the Finnish software industry.* The International Journal of Entrepreneurship and Innovation, 2007. **8**(2): p. 115-126.
29. Heirman, A. and B. Clarysse, *Which tangible and intangible assets matter for innovation speed in start-ups?* Journal of Product Innovation Management, 2007. **24**(4): p. 303-315.
30. de Haan, U. and S. Cohen, *The role of improvisation in Off-the-Shelf software development of entrepreneurial vendors.* 2007 International Conference on Systems Engineering and Modeling, Proceedings, 2007: p. 85-92.
31. Yingyu, D. and W. Ye. *The Mechanisms of Learning and the Survival of New Ventures*. 2008.
32. Chenoweth, S. *Undergraduate Software Engineering Students in Startup Businesses*. 2008: IEEE.
33. Ojala, A. and P. Tyrväinen, *Business models and market entry mode choice of small software firms.* Journal of International Entrepreneurship, 2006. **4**(2): p. 69-81.
34. Pauli, J.W., T.E. Lawrence, and B.F. Brown, *Development of a new software product from a classroom project.* Proceedings of the Fifth International Conference on Information Technology: New Generations, 2008: p. 97-100.
35. Shane, S. and D. Cable, *Network ties, reputation, and the financing of new ventures.* Management Science, 2002. **48**(3): p. 364-381.
36. Hung, S. and Y. Hsiao. *Mobilizing social capital to pursue entrepreneurship*. 2004.
37. Zahra, S., B. Matherne, and J. Carleton, *Technological resource leveraging and the internationalisation of new ventures.* Journal of International Entrepreneurship, 2003. **1**(2): p. 163-186.
38. Harrison, R., C. Mason, and P. Girling, *Financial bootstrapping and venture development in the software industry.* Entrepreneurship & Regional Development, 2004. **16**(4): p. 307-333.
39. Kollmann, T., *What is e-entrepreneurship? - fundamentals of company founding in the net economy.* International Journal of Technology Management, 2006. **33**(4): p. 322-340.
40. Tarnacha, A. and C.F. Maitland, *Entrepreneurship in mobile application development.* 2006 ICEC: Eighth International Conference on Electronic Commerce, Proceedings, 2006: p. 589-593.
41. Leem, C.S., H.S. Suh, and D.S. Kim, *A classification of mobile business models and its applications.* Industrial Management & Data Systems, 2004. **104**(1): p. 78-87.
42. Faber, E., et al. *Designing business models for mobile ICT services*. in *16th Electronic Commerce Conference*. 2003. Bled, Slovenia.
43. Gruber, M. and J. Henkel, *New ventures based on open innovation - an empirical analysis of start-up firms in embedded Linux.* International Journal of Technology Management, 2006. **33**(4): p. 356-372.
44. von Hippel, E. and G. von Krogh, *Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science.* Organization Science, 2003. **14**(2): p. 209-223.
45. Chesbrough, H., *Open innovation: The new imperative for creating and profiting from technology*. 2003: Harvard Business Press.
46. Aslett, M., *Open Source is not a Business Model*. 2008, the 451 Group.
47. Menzel, H.C., I. Aaltio, and J.M. Ulijn, *On the way to creativity: Engineers as intrapreneurs in organizations.* Technovation, 2007. **27**(12): p. 732-743.
48. Pinchot, G., *Intrapreneuring: why you don't have to leave the corporation to become an entrepreneur*. 1985: Harper & Row New York, NY.
49. Abetti, P.A., *The birth and growth of Toshiba's laptop and notebook computers: A case study in Japanese corporate venturing.* Journal of Business Venturing, 1997. **12**(6): p. 507-529.
50. Day, D., *Raising radicals: Different processes for championing innovative corporate ventures.* Organization Science, 1994. **5**(2): p. 148-172.

Complete list of sources

Abetti, P.A., The birth and growth of Toshiba's laptop and notebook computers: A case study in Japanese corporate venturing. Journal of Business Venturing, 1997. **12**(6): p. 507-529.

Aslett, M., Open Source is not a Business Model. 2008, the 451 Group.

Baron, J.N. and M.T. Hannan, Organizational Blueprints for Success in High-Tech Start-Ups: LESSONS FROM THE STANFORD PROJECT ON EMERGING COMPANIES. California Management Review, 2002. **44**(3): p. 8-36.

Bilen, S.G., et al., Developing and assessing students' entrepreneurial skills and mind-set. Journal of Engineering Education, 2005. **94**(2): p. 233-243.

Brem, A., The boundaries of innovation and entrepreneurship: conceptual background and essays on selected theoretical and empirical aspects, Ch 1. 2008: Gabler Verlag.

Chenoweth, S. Undergraduate Software Engineering Students in Startup Businesses. 2008: IEEE.

Chesbrough, H., Open innovation: The new imperative for creating and profiting from technology. 2003: Harvard Business Press.

Chesbrough, H., Open business models: How to thrive in the new innovation landscape. 2007: Harvard Business Press.

Companys, Y.E. and J.S. McMullen, Strategic entrepreneurs at work: The nature, discovery, and exploitation of entrepreneurial opportunities. Small Business Economics, 2007. **28**: p. 301-322.

Cougar, D., L. Higgins, and S. McIntyre. ``Differentiating creativity, innovation, and entrepreneurship for information service products and processes''. 1990.

Cusumano, M., The business of software: What every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad. 2004: Free Press.

Cusumano, M.A., Software in Ireland: A balance of entrepreneurship and... lifestyle management? Communications of the ACM, 2005. **48**(10): p. 25-27.

Day, D., Raising radicals: Different processes for championing innovative corporate ventures. Organization Science, 1994. **5**(2): p. 148-172.

de Haan, U. and S. Cohen, The role of improvisation in Off-the-Shelf software development of entrepreneurial vendors. 2007 International Conference on Systems Engineering and Modeling, Proceedings, 2007: p. 85-92.

Doboli, A., S. Doboli, and E. Currie. Preparing computer engineers for a global economy: a study on effective collaboration practices in global student teams. 2009: IEEE Press.

Dollinger, M.J., Entrepreneurship: strategies and resources. 1999, Upper Saddle River, New Jersey: Prentice hall.

Fitzgerald, B., The transformation of open source software. MIS Quarterly, 2006. **30**(3): p. 587-598.

Fortune, A. and H. Aldrich, Acquiring competence at a distance: Application service providers as a Hybrid Organizational Form. Journal of International Entrepreneurship, 2003. **1**(1): p. 103-119.

Gary, K., et al. Work-in-progress: Embedding entrepreneurship in the computing curricula. 2008.

Giarratana, M.S., The birth of a new industry: entry by start-ups and the drivers of firm growth - The case of encryption software. Research Policy, 2004. **33**(5): p. 787-806.

Giuri, P., F. Rullani, and S. Torrisi, Explaining leadership in virtual teams: The case of open source software. Information Economics and Policy, 2008. **20**(4): p. 305-315.

Gruber, M. and J. Henkel, New ventures based on open innovation - an empirical analysis of start-up firms in embedded Linux. International Journal of Technology Management, 2006. **33**(4): p. 356-372.

Harrison, R., C. Mason, and P. Girling, Financial bootstrapping and venture development in the software industry. Entrepreneurship & Regional Development, 2004. **16**(4): p. 307-333.

Heirman, A. and B. Clarysse, Which tangible and intangible assets matter for innovation speed in start-ups? Journal of Product Innovation Management, 2007. **24**(4): p. 303-315.

Hsu, D.H., E.B. Roberts, and C.E. Eesley, Entrepreneurs from technology-based universities: evidence from MIT. Research Policy, 2007. **36**(5): p. 768-788.

Hung, S. and Y. Hsiao. Mobilizing social capital to pursue entrepreneurship. 2004.

Igel, B. and N. Islam, Strategies for service and market development of entrepreneurial software designing firms. Technovation, 2001. **21**(3): p. 157-166.

J, K., Start Up, London: Warner.

Janson, M.A. and S. Wrycza, Information technology and entrepreneurship: three cases from Poland. International Journal of Information Management, 1999. **19**(5): p. 351-367.

Kaganer, E., S.D. Pawlowski, and S. Wiley-Patton, Building Legitimacy for IT Innovations: The Case of Computerized Physician Order Entry Systems. Journal of the Association for Information Systems. **11**(1): p. 1-33.

Kaplan, J., Start Up. 1994, London: Warner.

Kirsch, D., B. Goldfarb, and A. Gera, FORM OR SUBSTANCE: THE ROLE OF BUSINESS PLANS IN VENTURE CAPITAL DECISION MAKING. Strategic Management Journal, 2009. **30**(5): p. 487-515.

Kollmann, T., What is e-entrepreneurship? - fundamentals of company founding in the net economy. International Journal of Technology Management, 2006. **33**(4): p. 322-340.

Kuratko, D. and R. Hodgetts, Entrepreneurship: Theory, Process and Practice. 2004, Mason, Ohio: Thomson.

Latham, S., Contrasting Strategic Response to Economic Recession in Start-Up versus Established Software Firms. Journal of Small Business Management, 2009. **47**(2): p. 180-201.

Lechner, C., M. Dowling, and I. Welpe, Firm networks and firm development: The role of the relational mix. Journal of Business Venturing, 2006. **21**(4): p. 514-540.

Lee, R. and O. Jones, Networks, communication and learning during business start-up - The creation of cognitive social capital. International Small Business Journal, 2008. **26**(5): p. 559-594.

Lehrer, M., Science-driven vs. market-pioneering high tech: comparative German technology sectors in the late nineteenth and late twentieth centuries. Industrial and Corporate Change, 2005. **14**(2): p. 251-278.

Lerner, J. and J. Tirole, Some simple economics of open source. The journal of industrial economics, 2002. **50**(2): p. 197-234.

Lettl, C., K. Rost, and I. von Wartburg, Why are some independent inventors 'heroes' and others 'hobbyists'? The moderating role of technological diversity and specialization. Research Policy, 2009. **38**(2): p. 243-254.

Liberman-Yaconi, L., T. Hooper, and K. Hutchings, Toward a Model of Understanding Strategic Decision-Making in Micro-Firms: Exploring the Australian Information Technology Sector. Journal of Small Business Management. **48**(1): p. 70-95.

Lockett, N., et al., The influence of co-location in higher education institutions on small firms' perspectives of knowledge transfer. Entrepreneurship and Regional Development, 2009. **21**(3): p. 265-283.

Mangan, A. and S. Kelly, Information systems and the allure of organisational integration: a cautionary tale from the Irish financial services sector. European Journal of Information Systems, 2009. **18**(1): p. 66-78.

Mann, R. and T. Sager, Patents, venture capital, and software start-ups. Research Policy, 2007. **36**(2): p. 193-208.

McQuaid, R.W., Entrepreneurship and ICT industries: Support from regional and local policies. Regional Studies, 2002. **36**(8): p. 909-919.

Menzel, H.C., I. Aaltio, and J.M. Ulijn, On the way to creativity: Engineers as intrapreneurs in organizations. Technovation, 2007. **27**(12): p. 732-743.

Mourmant, G., M.J. Gallivan, and M. Kalika, Another road to IT turnover: the entrepreneurial path. European Journal of Information Systems, 2009. **18**(5): p. 498-521.

Mueller, T. and H. Gemunden, Founder team interaction, customer and competitor orientation in software ventures. Management Research News, 2009. **32**(6): p. 539-554.

Mustonen, M., Copyleft--the economics of Linux and other open source software. Information Economics and Policy, 2003. **15**(1): p. 99-121.

Nambisan, S., Software firm evolution and innovation-orientation. Journal of Engineering and Technology Management, 2002. **19**(2): p. 141-165.

Ojala, A. and P. Tyrväinen, Business models and market entry mode choice of small software firms. Journal of International Entrepreneurship, 2006. **4**(2): p. 69-81.

O'Mahony, S., Guarding the commons: how community managed software projects protect their work* 1. Research Policy, 2003. **32**(7): p. 1179-1198.

Osterwalder, A. and Y. Pigneur, Business Model Generation. 2010, Hoboken, New Jersey: Wiley.

Parthasarathy, B. and Y. Aoyama, From software services to R&D services: local entrepreneurship in the software industry in Bangalore, India. Environment and Planning A, 2006. **38**(7): p. 1269-1285.

Pauli, J.W., T.E. Lawrence, and B.F. Brown, Development of a new software product from a classroom project. Proceedings of the Fifth International Conference on Information Technology: New Generations, 2008: p. 97-100.

Pinchot, G., Intrapreneuring: why you don't have to leave the corporation to become an entrepreneur. 1985: Harper & Row New York, NY.

Plant, R., S. Wills, and C. Valle, Creative Entrepreneurship at Iconstruye: A Pan Andean e-Procurement Market Maker. Entrepreneurship Theory and Practice, 2008. **32**(3): p. 575-588.

Rae, D., Entrepreneurship: from Opportunity to Action. 2007, New York: Palgrave.

Rajala, R., J. Nissilä, and M. Westerlund, Revenue models in the open source software business. Handbook of Research on Open Source Software: Technological, Economic, and, 2007: p. 541.

Rajala, R., M. Rossi, and V. Tuunainen. A framework for analyzing software business models. 2003: Citeseer.

Rajala, R. and M. Westerlund, Business models a new perspective on firms' assets and capabilities: observations from the Finnish software industry. The International Journal of Entrepreneurship and Innovation, 2007. **8**(2): p. 115-126.

Read, S., et al., Effectual Entrepreneurship. 2011, London: Routledge.

Rusu, A. and R. Elliott, Work in progress: Smoothing the border between academic and professional software engineering environment through entrepreneurship. 36th Annual Frontiers in Education, Conference Program, Vols 1-4, 2006: p. 1591-1592.

Saee, J. and F. Benli, The role of marketing knowledge for Australian ICT based entrepreneurs in terms of their internationalization strategy. Proceedings of the 8th European Conference on Knowledge Management, Vol 1 and 2, 2007: p. 829-834.

Sambamurthy, V., A. Bharadwaj, and V. Grover, Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. MIS Quarterly, 2003. **27**(2): p. 237-263.

Sarasvathy, S., Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. Academy of management Review, 2001. **26**(2): p. 243-263.

Sarasvathy, S., Entrepreneurship as a science of the artificial. Journal of Economic Psychology, 2003. **24**(2): p. 203-220.

Sarasvathy, S., Making it happen: Beyond theories of the firm to theories of firm design. Entrepreneurship Theory and Practice, 2004. **28**(6): p. 519-531.

Sarasvathy, S., Effectuation: Elements of entrepreneurial expertise. 2008: Edward Elgar Publishing.

Sarasvathy, S. and N. Dew, New market creation through transformation. Journal of Evolutionary Economics, 2005. **15**(5): p. 533-565.

Shane, S. and D. Cable, Network ties, reputation, and the financing of new ventures. Management Science, 2002. **48**(3): p. 364-381.

Skinner, G., Investigation of technology-based entrepreneurship and issues with ICT innovation in Australia. Mmactee' 08: Proceedings of the 10th Wseas International Conference Mathermatical Methods and Computational Techniques in Electrical Engineering, 2008: p. 179-185.

Tarnacha, A. and C.F. Maitland, Entrepreneurship in mobile application development. 2006 ICEC: Eighth International Conference on Electronic Commerce, Proceedings, 2006: p. 589-593.

Varis, J., O. Kuivalainen, and S. Saarenketo, Partner selection for international marketing and distribution in corporate new ventures. Journal of International Entrepreneurship, 2005. **3**(1): p. 19-36.

Vissa, B. and A. Chacar, Leveraging ties: the contingent value of entrepreneurial teams' external advice networks on Indian software venture performance. Strategic Management Journal, 2009. **30**(11): p. 1179-1191.

Waguespack, D.M. and L. Fleming, Scanning the Commons? Evidence on the Benefits to Startups Participating in Open Standards Development. Management Science, 2009. **55**(2): p. 210-223.

Wang, P. and E. Swanson, Launching professional services automation: Institutional entrepreneurship for information technology innovations. Information and organization, 2007. **17**(2): p. 59-88.

West, J., How open is open enough?: Melding proprietary and open source platform strategies. Research Policy, 2003. **32**(7): p. 1259-1285.

Wiltbank, R., et al., What to do next? The case for non predictive strategy. Strategic Management Journal, 2006. **27**(10): p. 981-998.

Yingyu, D. and W. Ye. The Mechanisms of Learning and the Survival of New Ventures. 2008.

Zackariasson, P. and T. Wilson. Game on: Competition and competitiveness in the video game industry. 2008.

Zahra, S., B. Matherne, and J. Carleton, Technological resource leveraging and the internationalisation of new ventures. Journal of International Entrepreneurship, 2003. **1**(2): p. 163-186.

Zahra, S.A. and W.C. Bogner, Technology strategy and software new ventures' performance: Exploring the moderating effect of the competitive environment. Journal of Business Venturing, 2000. **15**(2): p. 135-173.